The original link to the blog post can be found here: https://www.optiv.com/insights/source-zero/blog/redirectors-red-teamers-introduction

November 20, 2023

Redirectors: A Red Teamer's Introduction

Introduction

For as long as offensive computer network operations have occurred, whether performed by malicious actors or security consultants, attackers have made attempts to keep their infrastructure hidden and protected. At the most fundamental level, this typically involves the redirection of network traffic from the target's network through a host or service that obscures the traffic's final destination. In this blog post, we will take a high-level look at the purpose of redirectors and discuss their implementation.

Please note that the examples discussed represent the tip of the iceberg in terms of redirection, and this blog post should be considered a non-exhaustive primer.

Overview

Put simply, redirectors are systems that **proxy traffic from a target network to private attack infrastructure**. They serve multiple purposes, including to:

- Prevent unauthorized access to attack infrastructure
- Control the appearance of the infrastructure to the client
- Protect the attack chain

Below is more information about these different purposes.

Prevent unauthorized access to the attack infrastructure:

Redirectors act as the demarcation line between the public internet and private attack infrastructure. Private attack infrastructure may include phishing or Command and Control (C2) servers, decoy web servers, staging servers, or any other component that must remain protected from unauthorized parties. Redirectors provide granular control over access to private attack infrastructure, allowing only the desired traffic to pass through.

Control the appearance of the infrastructure:

Given that a redirector establishes a boundary between the public internet and private attack infrastructure, it often is the only publicly visible infrastructure component and the nearest hop to the target environment. The redirector's characteristics and appearance inform the target, and anyone who investigates the host, of the purpose and use of the host. Redirectors offer an opportunity to shape the perception and to deceive or mislead.

Protect the attack chain:

Redirectors are considered burnable components of attack infrastructure. If a redirector is identified and categorized as malicious, then it can be burned down and replaced with a new redirector. The new redirector can then be connected to the existing private attack infrastructure. The attack chain, although impacted by the burning of a domain, IP, or redirector, remains mostly intact, as the core components are hidden behind the redirector.

By segmenting and controlling traffic to private attack infrastructure via the use of redirectors, the more time-consuming and sensitive components to set up—such as C2 or phishing servers—remain protected and inaccessible to defenders.

Redirector Types

For the purposes of this blog post, let's think of basic redirector types as existing on a spectrum of less to more filtering. Generally, this also correlates to less or more effort to configure and deploy. However, the level of effort on either end of the spectrum can be diminished through automation.

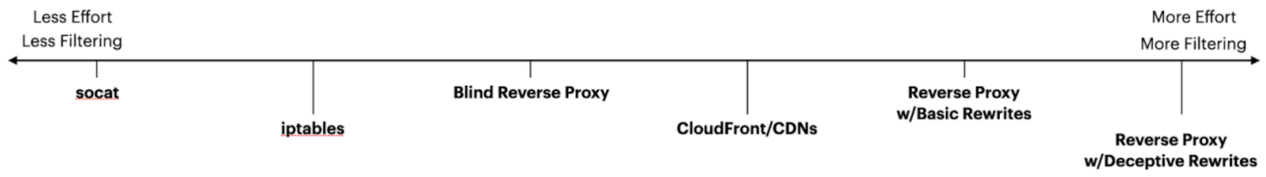


Figure 1 - Redirector Types

On the "less effort" and "less filtering" end of the spectrum, we have redirectors based on socat, iptables, or blind reverse proxies. These types of redirectors generally tend not to filter or apply rules to traffic. Instead, they blindly forward, or bend, traffic to a destination. Setup involves relatively little effort. However, there are implications related to the level of operational security they offer. With few filters, these redirectors will pass most traffic, including from Internet scanners or mail filtering solutions, right through to backend servers. Depending on the use case, this may be acceptable. However, it may increase the likelihood of identification and categorization of the redirector and/or backend assets as malicious.

On the "more effort" and "more filtering" end of the spectrum, we have redirectors built using reverse proxies that allow for filtering traffic based on specific conditions and rules. These redirectors may only pass traffic with specific User Agent strings to a backend C2 server, or they may redirect traffic from pre-defined IP ranges to a decoy web server. The intricacies of the configuration depend on the engagement requirements, the capabilities of the reverse proxy and beacon being used, and, lastly, the engineer's imagination. More filters mean more control over how traffic is processed, which, if configured correctly, may improve the protection of infrastructure and decrease the likelihood that it is identified as malicious.

Implementation

There are many reverse proxies available to choose from, each with their own advantages and disadvantages. The following implementation will use Apache, a web server featuring robust reverse proxy configuration options. After describing the basic setup with Apache, we will add a CloudFront distribution in front of the redirector to demonstrate an additional layer of redirection. For the C2 framework, this example will use Cobalt Strike. However, the basic redirector configuration is generally C2 framework agnostic, and you may use the framework of your choosing. Please note that discussions of the C2 deployment and configuration are not in scope for this post.

Before getting started, there are several requirements for this demonstration:

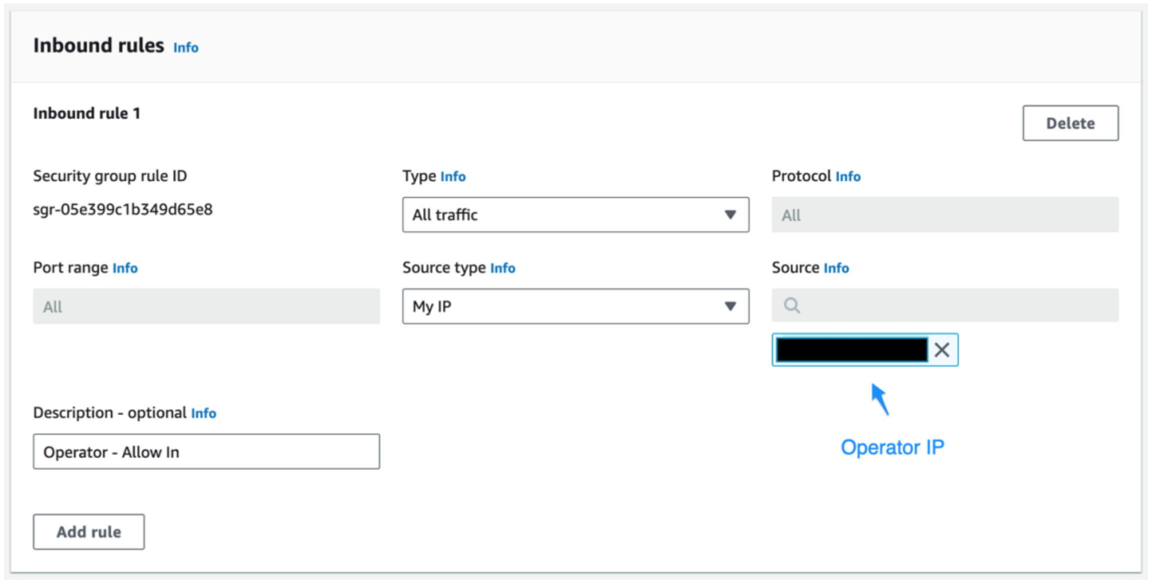
- An Amazon Web Services (AWS) account with the ability to deploy EC2s and create CloudFront distributions
- At least one domain with the ability to set the domain's A Record
- C2 team server software such as Cobalt Strike, Sliver, etc.

For the purposes of this demonstration, we are setting up a Cobalt Strike team server on an AWS EC2 instance. However, given that during live engagements, team servers can end up hosting data from the target network. It is preferrable that the team server be hosted on a server owned by your organization.

EC2 Provisioning

Using your preferred method, whether via the AWS console, Terraform or otherwise, provision two EC2 instances of the Linux distribution of your choosing: one for the redirector and one for the team server.

Create a security group for each EC2 instance, as we will apply different rules for each resource. Initially, configure the security groups to allow traffic inbound *only* from the operator's IP.


Inbound rules Info

Inbound rule 1

Delete

Security group rule ID: sgr-05e399c1b349d65e8

Type Info: All traffic

Protocol Info: All

Port range Info: All

Source type Info: My IP

Source Info: [redacted]

Description - optional Info: Operator - Allow In

Add rule

Figure 2 - Initial Redirector Security Group Rules

Apache Installation and Initial Configuration

On the redirector EC2 instance, install Apache along with several Apache modules that will be required for the configuration:

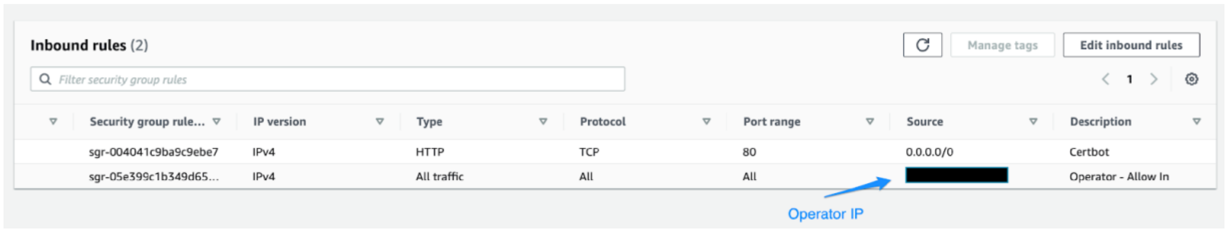
```
apt-get install apache2
systemctl stop apache2.service
a2enmod ssl proxy remoteip rewrite proxy_http
```

Certbot Installation and Execution

Certbot is a tool that obtains TLS certificates from Let's Encrypt. We will use Certbot's HTTP-01 Challenge type to obtain a certificate, which requires that:

- The A Record for your domain points to the IP address where Certbot is being executed
- Certbot is able to start up and expose a web server (hence disabling the Apache service above), as well as receive inbound HTTP traffic

Before installing and running Certbot, modify the redirector security group rules to allow inbound traffic from anywhere to port 80.

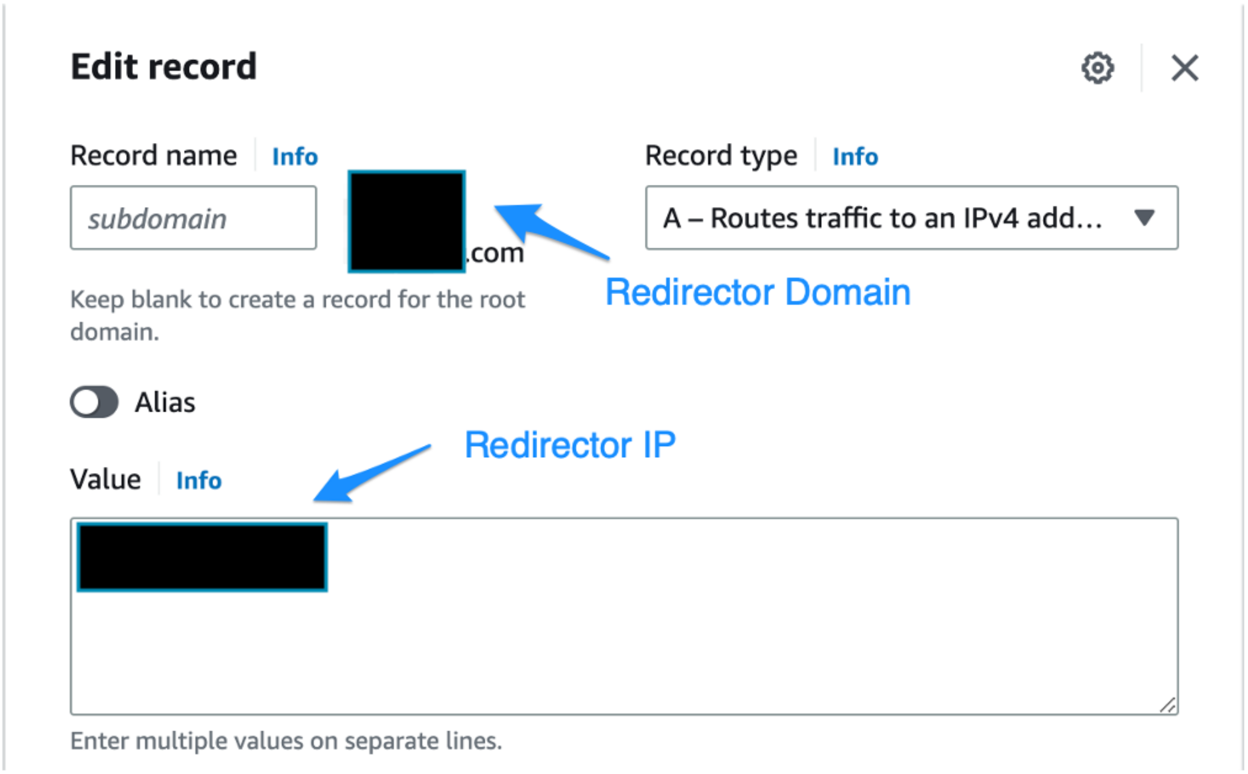


Inbound rules (2)

	Security group rule...	IP version	Type	Protocol	Port range	Source	Description
	sgr-004041c9ba9c9ebe7	IPv4	HTTP	TCP	80	0.0.0.0/0	Certbot
	sgr-05e399c1b349d65...	IPv4	All traffic	All	All	[redacted]	Operator - Allow In

Figure 3 - Security Group Modification Prior to Running Certbot

Next, add the IP address of the redirector as the A Record for the domain being used for the redirector.



Edit record

Record name: subdomain [redacted].com

Keep blank to create a record for the root domain.

Record type: A – Routes traffic to an IPv4 add...

Alias

Value: [redacted]

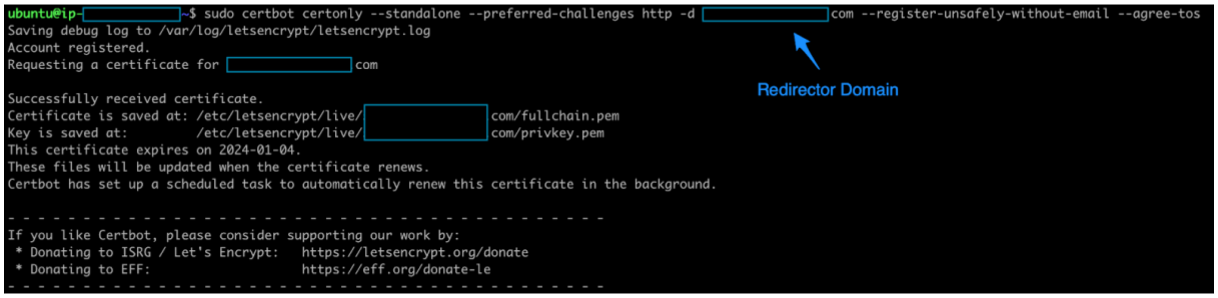
Enter multiple values on separate lines.

Figure 4 - A Record Entry for Redirector Domain

Once those steps are complete, install and run Certbot by executing the following:

```
apt-get install certbot
# replace example.com with your domain
certbot certonly --standalone --preferred-challenges http -d
exampledomain.com
--register-unsafely-without-email --agree-tos
```

Given a successful Certbot run, a response similar to the following screenshot should be displayed.



```
ubuntu@ip-[redacted]~$ sudo certbot certonly --standalone --preferred-challenges http -d [redacted]com --register-unsafely-without-email --agree-tos
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Account registered.
Requesting a certificate for [redacted]com

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/[redacted]com/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/com/privkey.pem
This certificate expires on 2024-01-04.
These files will be updated when the certificate renews.
Certbot has set up a scheduled task to automatically renew this certificate in the background.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Figure 5 - Certbot Run

Once the certificate files have been obtained, remove the AWS security group rule that permitted traffic inbound on port 80.

Apache Site Configuration

The heart of the redirector configuration is found in the Apache site configuration file. The site file defines the desired behavior for network traffic that reaches the redirector as well as the TLS configuration, specifying which certificate files to use and whether to perform TLS verification.

Within the /etc/apache/sites-available/ directory, create a configuration file called domain.com.conf where domain is the name of the domain being used for your redirector.

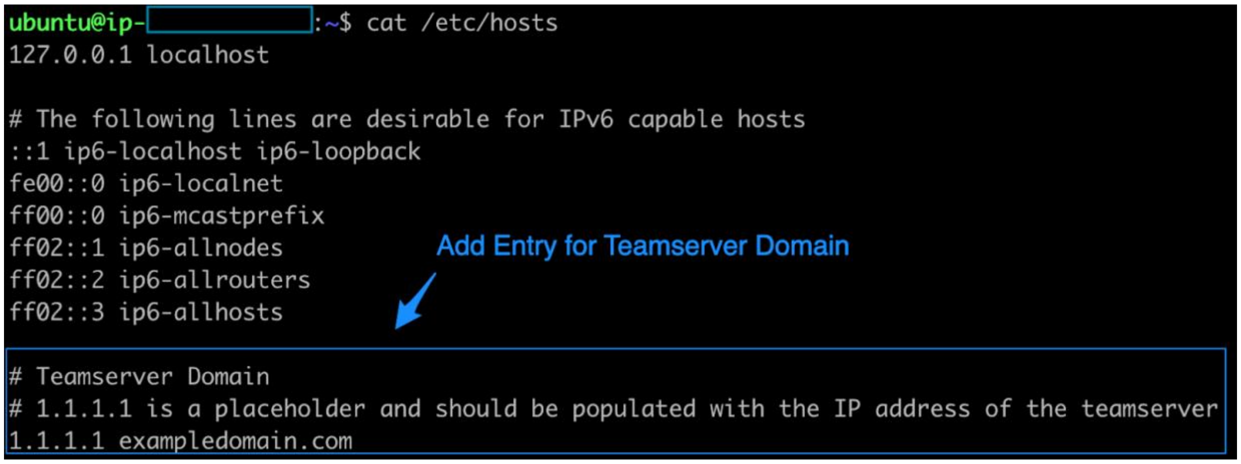
The first step is to redirect unencrypted inbound traffic on port 80 to port 443. Using Apache's VirtualHost directive, define the following block (replace exampledomain.com with the domain being used for the redirector):

```
<VirtualHost *:80>
ServerName exampledomain.com
RewriteEngine On
RewriteCond %{HTTPS} off
RewriteRule ^(.*)$ https://%{HTTP_HOST}%{REQUEST_URI} [L,R=301]


ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

The above block will redirect traffic to port 443, which will be handled by the block below. In the following VirtualHost directive, Apache is instructed to proxy all traffic to "exampledomain.com," or the domain you selected for your redirector. In order for this to pass traffic to the team server domain, we will need to modify the /etc/hosts file on the redirector to resolve "exampledomain.com" to the IP address of the team server. We are using this method because it allows us to reuse the TLS certificates obtained via Certbot instead of needing a second domain and second set of certificates.

```
<VirtualHost *:443>
ServerName exampledomain.com
RemoteIPHeader X-Forwarded-For

ProxyPass / https://exampledomain.com/
ProxyPassReverse / https://exampledomain.com/

SSLProxyEngine On
SSLProxyVerify none
SSLProxyProtocol TLSv1.2

SSLCertificateFile /etc/letsencrypt/live/exampledomain.com/fullchain.pem
SSLCertificateKeyFile /etc/letsencrypt/live/exampledomain.com/privkey.pem

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Next, add an entry to /etc/hosts pointing DNS lookups for your domain to the IP of the team server. When Apache uses the ProxyPass directives to proxy the traffic to the specified domain, it will then reference the redirector's /etc/hosts file.



```
ubuntu@ip-[redacted]:~$ cat /etc/hosts
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts

# Teamserver Domain
# 1.1.1.1 is a placeholder and should be populated with the IP address of the teamserver
1.1.1.1 exampledomain.com
```

Figure 6 - Example Hosts File

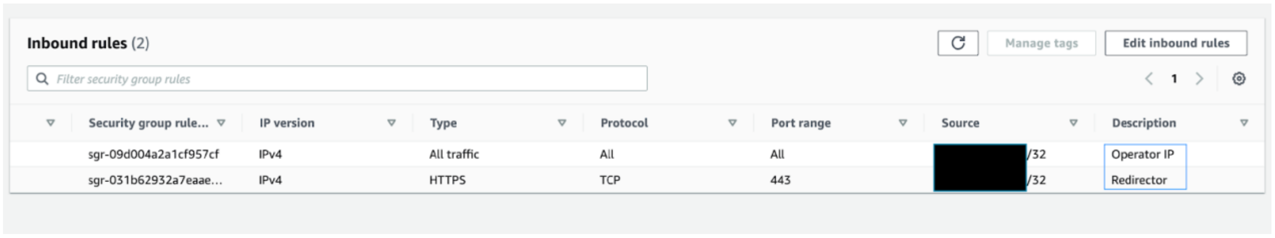
Then, enable the site and restart the Apache service:

```
a2ensite domain.com
systemctl restart apache2.service
```

Command and Control Server Configuration

At the beginning of the walkthrough, we created two EC2s, including one for the redirector and one for the team server. This blog post assumes that the reader can successfully install and configure the C2 framework of their choosing. For the purposes of this demonstration, we have deployed a team server running Cobalt Strike.

Prior to opening the firewall rules to allow traffic in from the redirector, the certificate material from the redirector must be copied to the team server. Depending on your C2 framework, the certificate fullchain and private key files may be added to your C2 configuration in different ways. Once the certificate material has been added, modify the team server security group rules to allow HTTPS traffic inbound from the redirector IP.

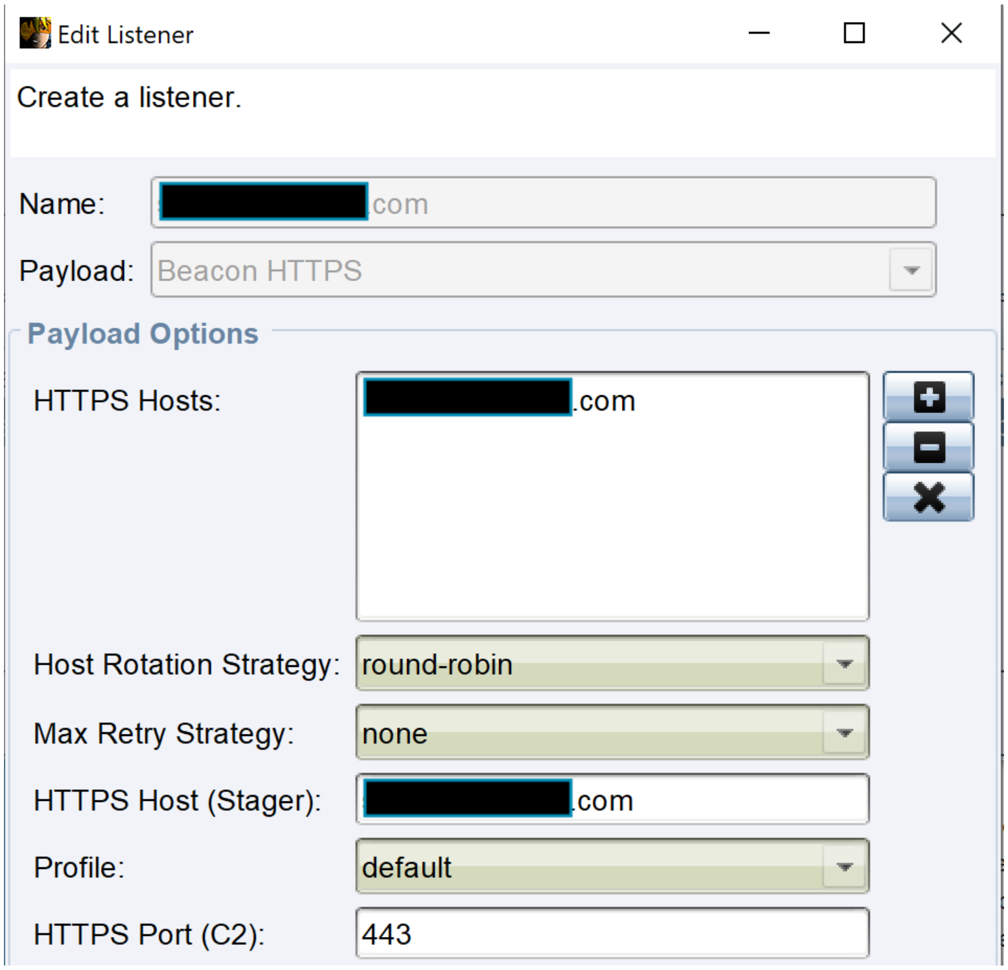


Inbound rules (2)

Security group rule...	IP version	Type	Protocol	Port range	Source	Description
sgr-09d004a2a1cf957cf	IPv4	All traffic	All	All	[redacted]/32	Operator IP
sgr-031b62932a7eaae...	IPv4	HTTPS	TCP	443	[redacted]/32	Redirector

Figure 7 – Team Server Security Group Rules

Using your C2 framework, create a listener and payload that are configured to use the previously configured redirector domain name as the HTTPS host.



Edit Listener

Create a listener.

Name:	[redacted].com
Payload:	Beacon HTTPS

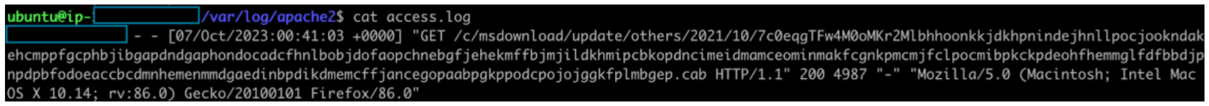
Payload Options

HTTPS Hosts:	[redacted].com
Host Rotation Strategy:	round-robin
Max Retry Strategy:	none
HTTPS Host (Stager):	[redacted].com
Profile:	default
HTTPS Port (C2):	443

Figure 8 - Cobalt Strike Listener

Assuming that you are executing the test payload on a Windows VM running on your host, there is no need to add additional firewall rules. However, if you are executing the payload from another workstation with a different IP, adjust the security group rules accordingly.

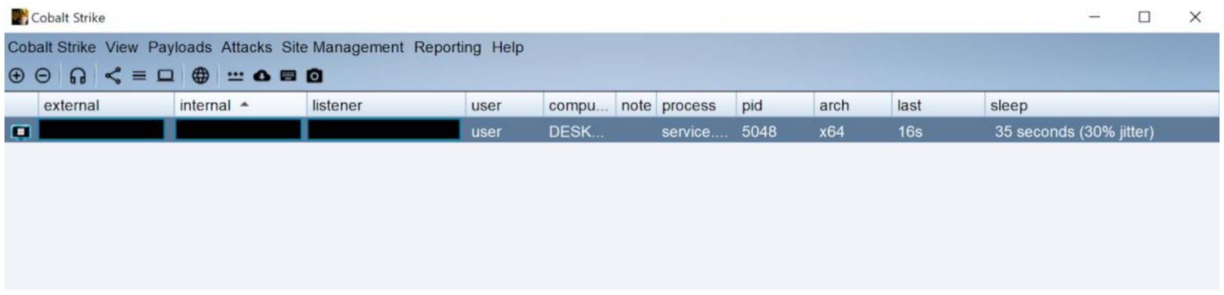
Once the payload has been executed on the redirector, there should now be an entry in the Apache access.log file indicating that an HTTP request has been made.



```
ubuntu@ip-[redacted]/var/log/apache2$ cat access.log
[redacted] - - [07/Oct/2023:00:41:03 +0000] "GET /c/msdownload/update/others/2021/10/7c0eqgTFw4M0oMKr2Mlbhhoonkkjdkhpnindejhnllpocjookndak
ehcmppfgcphbjibgapdndgaphondocadcfhnlbobjdofaopchnebgfjehekmffbjmjildkhmipcbkopdncimeidmamceominmakfcgnkpmcmjfclpocmibpkckpdeohfhemmglfdfbbdjp
npdpbfodoeaccbcdmnhemenmmdgaedinbpdikdmemcffjancegopaabpgkppodcpojojggkfplmbgep.cab HTTP/1.1" 200 4987 "-" "Mozilla/5.0 (Macintosh; Intel Mac
OS X 10.14; rv:86.0) Gecko/20100101 Firefox/86.0"
```

Figure 9 - Apache Access Log on Redirector

On your C2 team server, you should see an established session. If the C2 framework being used supports the X-Forwarded-For header, then the client's origin IP should be available.


Cobalt Strike
Cobalt Strike View Payloads Attacks Site Management Reporting Help

	external	internal ▲	listener	user	compu...	note	process	pid	arch	last	sleep
	[redacted]	[redacted]	[redacted]	user	DESK...		service....	5048	x64	16s	35 seconds (30% jitter)

Figure 10 - Cobalt Strike Beacon Callback

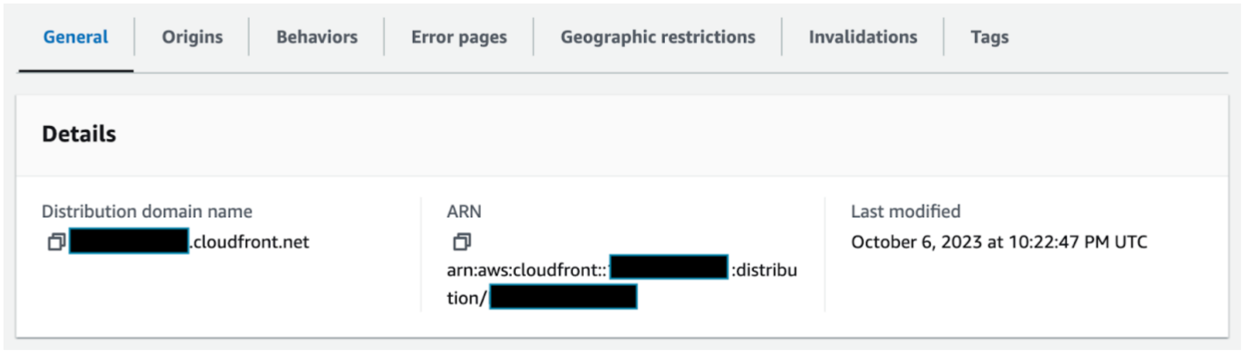
To summarize, a payload has been executed, a request has been made to the domain hosted on the redirector, and, finally, the request has been passed through the reverse proxy on the redirector to the backend team server. The team server only allows traffic in from the redirector and from operator IPs, thus making it inaccessible to prying eyes. This setup provides the team server with some degree of protection, although it does not filter out unwanted traffic from reaching the team server.

CloudFront Redirection

As an added layer of redirection, and to help with issues related to domain categorization, content delivery networks (CDNs) can be used to redirect traffic. Building on the example just discussed, CloudFront can be easily integrated without any adjustments to the redirector configuration itself. In general, there are two steps to take:

- Create and configure a CloudFront distribution, as illustrated in this blog post, that points to the previously configured domain

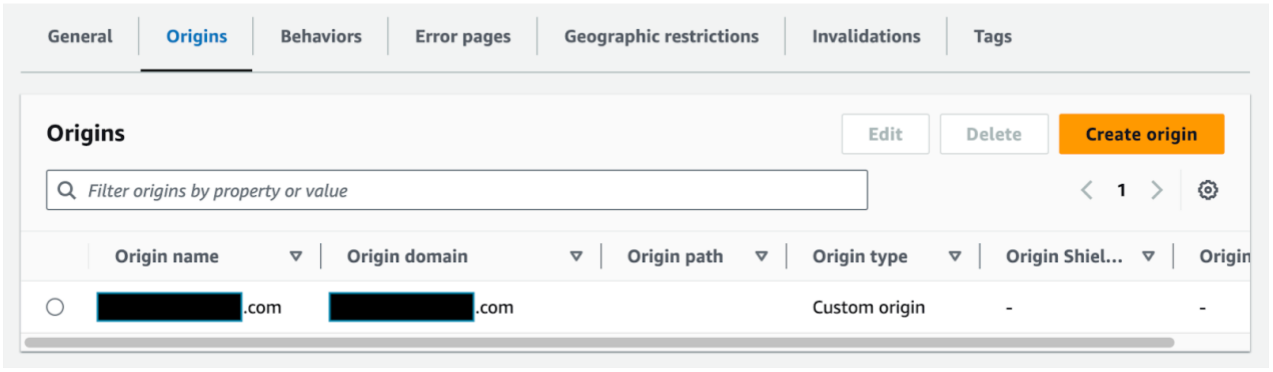
- Modify the C2 profile to include the CloudFront distribution URL instead of the domain



Details

Distribution domain name	ARN	Last modified
[redacted].cloudfront.net	arn:aws:cloudfront::[redacted]:distribution/[redacted]	October 6, 2023 at 10:22:47 PM UTC

Figure 11 - CloudFront Distribution Details

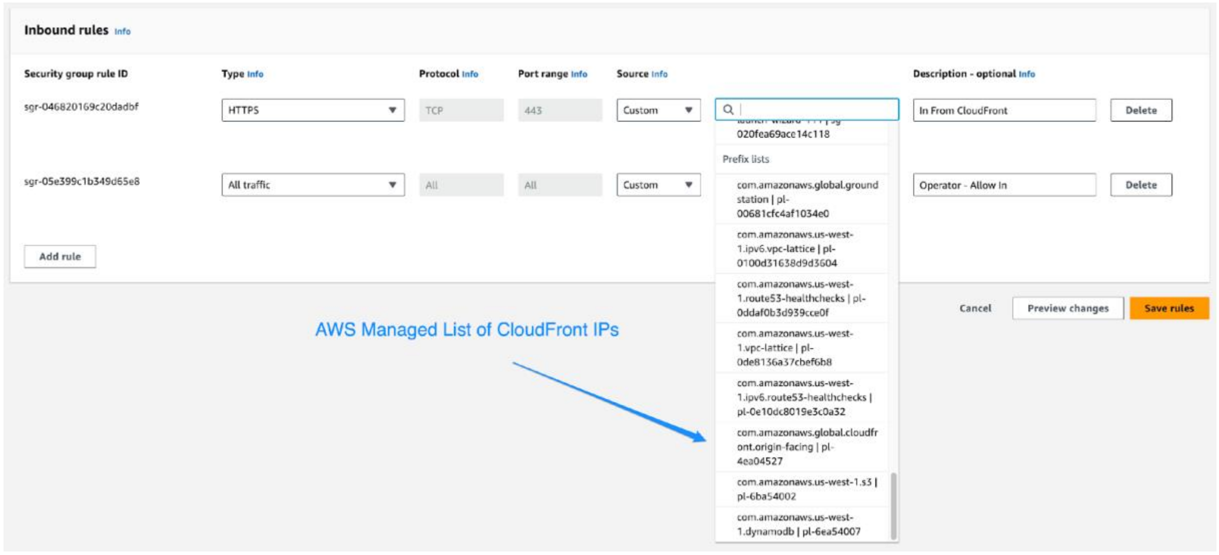


Origins

Origin name	Origin domain	Origin path	Origin type	Origin Shiel...	Origin
[redacted].com	[redacted].com		Custom origin	-	-

Figure 12 - CloudFront Distribution Origins

Before generating and executing a new payload, the firewall rules on the redirector need to be modified to allow traffic in from CloudFront. Within AWS security groups, there is a feature that enables users to add firewall rules for dynamically managed lists of resources that AWS maintains. Specifically useful in this scenario is a managed prefix list for all CloudFront IPs. This list enables operators to allow HTTPS traffic in from a dynamic list of CloudFront IPs, while preventing other traffic from reaching the redirector.

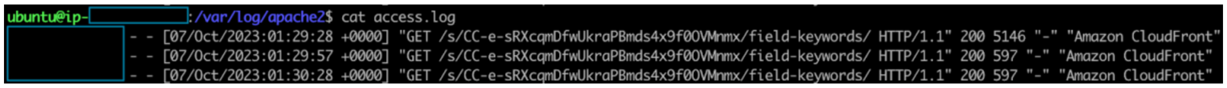


Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-046820169c20dadbf	HTTPS	TCP	443	Custom		In From CloudFront	Delete
sgr-05e399c1b349d65e8	All traffic	All	All	Custom		Operator - Allow In	Delete

```
020fea69ace14c118
Prefix lists
com.amazonaws.global.groundstation | pl-00681cfc4af1034e0
com.amazonaws.us-west-1.ipv6.vpc-lattice | pl-0100d31638d9d3604
com.amazonaws.us-west-1.route53-healthchecks | pl-0ddaf0b3d939cce0f
com.amazonaws.us-west-1.vpc-lattice | pl-0de8136a37cbef6b8
com.amazonaws.us-west-1.ipv6.route53-healthchecks | pl-0e10dc8019e3c0a32
com.amazonaws.global.cloudfront.origin-facing | pl-4ea04527
com.amazonaws.us-west-1.s3 | pl-6ba54002
com.amazonaws.us-west-1.dynamodb | pl-6ea54007
```

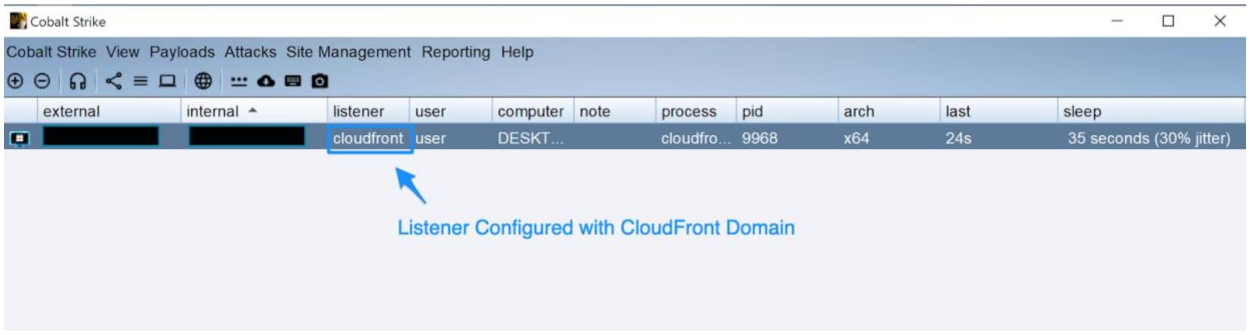
Figure 13 - AWS Managed Prefix

Once the appropriate managed prefix has been added to the firewall rules, generate and execute the new payload using the CloudFront URL as the C2 URI. If configured correctly, the Apache access.log on the redirector should show traffic from CloudFront and the team server should show a new beacon.



```
ubuntu@ip-[redacted]:/var/log/apache2$ cat access.log
[redacted] - - [07/Oct/2023:01:29:28 +0000] "GET /s/CC-e-sRXcqmDfwUkraPBmds4x9f00VMnmx/field-keywords/ HTTP/1.1" 200 5146 "-" "Amazon CloudFront"
           - - [07/Oct/2023:01:29:57 +0000] "GET /s/CC-e-sRXcqmDfwUkraPBmds4x9f00VMnmx/field-keywords/ HTTP/1.1" 200 597 "-" "Amazon CloudFront"
           - - [07/Oct/2023:01:30:28 +0000] "GET /s/CC-e-sRXcqmDfwUkraPBmds4x9f00VMnmx/field-keywords/ HTTP/1.1" 200 597 "-" "Amazon CloudFront"
```

Figure 14 - Apache Access Log on Redirector


Cobalt Strike
Cobalt Strike View Payloads Attacks Site Management Reporting Help
Listener Configured with CloudFront Domain

external	internal ▲	listener	user	computer	note	process	pid	arch	last	sleep
[redacted]	[redacted]	cloudfront	user	DESKT...		cloudfro...	9968	x64	24s	35 seconds (30% jitter)

Figure 15 - Cobalt Strike Beacon Callback via CloudFront Domain

Take note that due to the addition of the X-Forwarded-For header on the redirector, the external IP of the beacon displays the real client origin IP.

To summarize, a payload has been executed, a request has been made to a CloudFront distribution, the CloudFront distribution has forwarded the request to the specified domain, and the request has been passed through the reverse proxy on the redirector to the backend team server.

Next Steps

The redirector setup discussed in this post is about as simple as it can get outside of using socat or iptables. From here, an operator could add a myriad of rules defining the paths, User-Agent strings, or other characteristics that are required for traffic to be passed to the backend team server. An operator could also build in deceptive rewrites that seek to control the narrative around the purpose of the exposed infrastructure. For further information on associated TTPs, please reference the following resources:

- HTTPS Payload and C2 Redirectors
- Modern Red Team Infrastructure
- Red Team Infrastructure Done Right