



Features Business Explore Marketplace Pricing

This repository

Search

Sign in or Sign up

 **bluscreenofjeff / Red-Team-Infrastructure-Wiki**

 Watch

124

 Star

1,093

 Fork

252

 Code

 Issues **1**

 Pull requests **0**

 Projects **0**

 Insights

## Join GitHub today

GitHub is home to over 20 million developers working together to host and review code, manage projects, and build software together.

Sign up

Dismiss

Wiki to collect Red Team infrastructure hardening resources

infrastructure

redirector

cobalt-strike

empire

red-team

pentesting

 **105** commits

 **1** branch

 **0** releases

 **9** contributors

 BSD-3-Clause

Branch: **master** ▾

New pull request

Find file

Clone or download ▾



**bluscreenofjeff** Merge pull request [#26](#) from Chiggins/master ...

Latest commit 8fe6c65 25 days ago

 [images](#)

Added more to the phishing section

9 months ago

## README.md

This wiki is intended to provide a resource for setting up a resilient Red Team infrastructure. It was made to complement Steve Borosh ([@424f424f](#)) and Jeff Dimmock's ([@bluscreenofjeff](#)) BSides NoVa 2017 talk "Doomsday Preppers: Fortifying Your Red Team Infrastructure" ([slides](#))

If you have an addition you'd like to make, please submit a Pull Request or file an issue on the repo.

THANK YOU to all of the authors of the content referenced in this wiki and to all who [contributed](#)!

# Table of Contents

---

- [Design Considerations](#)
  - [Functional Segregation](#)
  - [Using Redirectors](#)
  - [Sample Design](#)
  - [Further Resources](#)
- [Domains](#)
  - [Categorization and Blacklist Checking Resources](#)
- [Phishing](#)
  - [Easy Web-Based Phishing](#)
  - [Cobalt Strike Phishing](#)

- [Phishing Frameworks](#)
- [Redirectors](#)
  - [SMTP](#)
    - [Sendmail](#)
      - [Remove previous server headers](#)
      - [Configure a catch-all address](#)
    - [Postfix](#)
  - [DNS](#)
    - [socat for DNS](#)
    - [iptables for DNS](#)
  - [HTTP\(S\)](#)
    - [socat vs mod\\_rewrite](#)
    - [socat for HTTP](#)
    - [iptables for HTTP](#)
    - [ssh for HTTP](#)
    - [Payloads and Web Redirection](#)
    - [C2 Redirection](#)
      - [C2 Redirection with HTTPS](#)
    - [Other Apache mod\\_rewrite Resources](#)
- [Modifying C2 Traffic](#)
  - [Cobalt Strike](#)
  - [Empire](#)
- [Third-Party C2 Channels](#)
  - [Domain Fronting](#)
    - [Further Resources on Domain Fronting](#)

- [PaaS Redirectors](#)
- [Other Third-Party C2](#)
- [Obscuring Infrastructure](#)
- [Securing Infrastructure](#)
- [Automating Deployments](#)
- [General Tips](#)
- [Thanks to Contributors](#)

## Design Considerations

---

### Functional Segregation

---

When designing a red team infrastructure that needs to stand up to an active response or last for a long-term engagement (weeks, months, years), it's important to segregate each asset based on function. This provides resilience and agility against the Blue Team when campaign assets start getting detected. For example, if an assessment's phishing email is identified, the Red Team would only need to create a new SMTP server and payload hosting server, rather than a whole team server setup.

Consider segregating these functions on different assets:

- Phishing SMTP
- Phishing payloads
- Long-term command and control (C2)
- Short-term C2

Each of these functions will likely be required for each social engineering campaign. Since active incident response is typical in a Red Team assessment, a new set of infrastructure should be implemented for each campaign.

## Using Redirectors

---

To further resilience and concealment, every back-end asset (i.e. team server) should have a redirector placed in front of it. The goal is to always have a host between our target and our backend servers. Setting up the infrastructure in this manner makes rolling fresh infrastructure much quicker and easier - no need to stand up a new team server, migrate sessions, and reconnect non-burned assets on the backend.

Common redirector types:

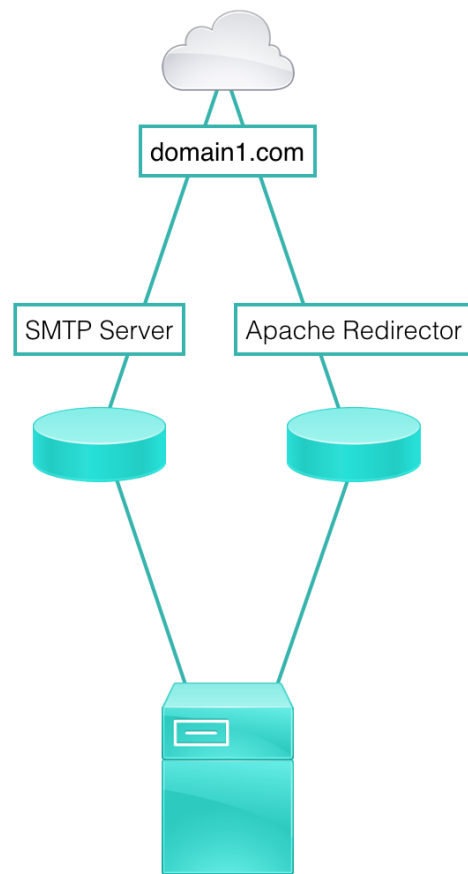
- SMTP
- Payloads
- Web Traffic
- C2 (HTTP(S), DNS, etc)

Each redirector type has multiple implementation options that best fit different scenarios. These options are discussed in further detail in the [Redirectors](#) section of the wiki. Redirectors can be VPS hosts, dedicated servers, or even apps running on a Platform-as-a-Service instance.

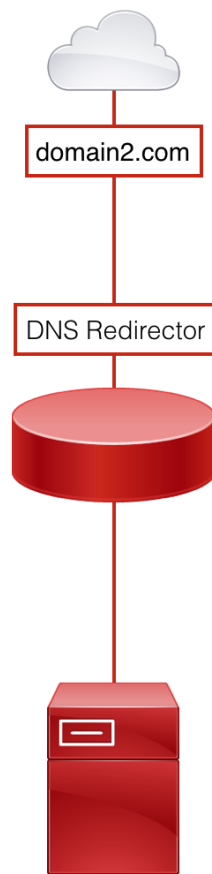
## Sample Design

---

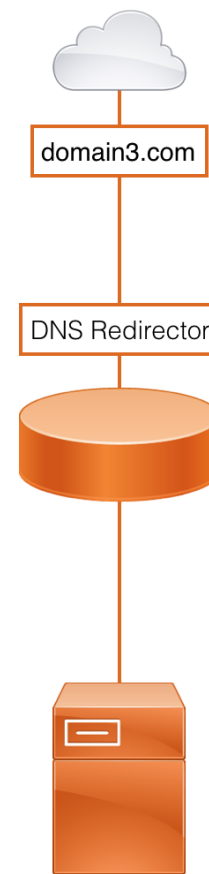
Here is a sample design, keeping functional segregation and redirector usage in mind:



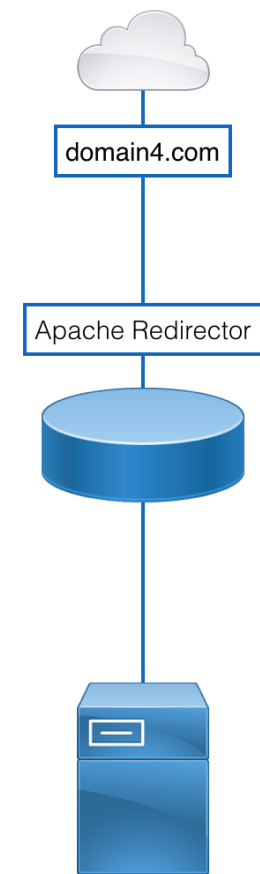
Phishing &  
Payloads



LT DNS C2



ST DNS C2



ST HTTP C2

## Further Resources

- [A Vision for Distributed Red Team Operations - Raphael Mudge \(@armitagehacker\)](#)
- [Infrastructure for Ongoing Red Team Operations - Raphael Mudge](#)

- [Advanced Threat Tactics \(2 of 9\): Infrastructure - Raphael Mudge](#)
- [Cloud-based Redirectors for Distributed Hacking - Raphael Mudge](#)
- [6 Red Team Infrastructure Tips - Alex Rymdeko-Harvey \(@killswitch-gui\)](#)
- [How to Build a C2 Infrastructure with Digital Ocean – Part 1 - Lee Kagan \(@invokethreatguy\)](#)
- [Automated Red Team Infrastructure Deployment with Terraform - Part 1 - Rasta Mouse \(@\\_RastaMouse\)](#)

## Domains

---

Perceived domain reputation will vary greatly depending on the products your target is using, as well as their configuration. As such, choosing a domain that will work on your target is not an exact science. Open source intelligence gathering (OSINT) will be critical in helping make a best guess at the state of controls and which resources to check domains against. Luckily, online advertisers face the same problems and have created some solutions we can leverage.

[expirddomains.net](#) is a search engine for recently expired or dropped domains. It provides search and advanced filtering, such as age of expiration, number of backlinks, number of Archive.org snapshots, [SimilarWeb](#) score. Using the site, we can register pre-used domains, which will come with domain age, that look similar to our target, look similar to our impersonation, or simply are likely to blend in on our target's network.



Expired Domains

Deleted Domains

Domain Lists

Links

## Domain Name Search



google



Show Filter (About 49 Domains)

Next Page »

Domain	BL	DP	ABY	ACR	SimilarWeb	STC	Dmoz	C	N	O	D	TLDs Reg	SG	CO	List	Status	RL
zarobotayka-v-google.ru	0	0	-	0	13.4 M	87%	-	✓	✓	✓	✓	0	0	0	Expired	Register	
ComGoogle.com.br	0	0	2013	4	13.5 M	100%	-	✓	✓	✓	✓	14	33.1 K	9	Expired	Register	
googlew.com.au	0	0	2009	11	14.4 M	100%	-	✓	✓	✓	✓	16	110.0 K	1	Expired	Register	
GoogleGadgetBlog.com	11	3	2009	30	16.6 M	100%	-	✓	✓	✓	✓	0	40	18	Expired	Register	
ipv6google.com	4	1	2009	11	16.7 M	100%	-	✓	✓	✓	✓	0	20	6	Expired	Register	
ItGoogle.it	1	0	2007	11	19.0 M	100%	-	✓	✓	✓	✓	4	480	4	Expired	Register	
GooglePlus.com.mx	0	0	2013	2	19.7 M	100%	-	✓	✓	✓	✓	27	2.2 M	1	Expired	Register	
YouGoogleTube.com	1	0	2007	6	20.8 M	100%	-	✓	✓	✓	✓	0	10	1	Expired	Register	
www0google.fr	1	0	2007	11	21.5 M	56%	-	✓	✓	✓	✓	1	320	0	Expired	Register	
googlefotos.com.br	0	0	-	0	21.8 M	100%	-	✓	✓	✓	✓	3	550.0 K	9	Expired	Register	

When choosing a domain for C2 or data exfiltration, consider choosing a domain categorized as Finance or Healthcare. Many organizations will not perform SSL middling on those categories due to the possibility of legal or data sensitivity issues. It is also important to ensure your chosen domain is not associated with any previous malware or phishing campaigns.

The tool [CatMyFish](#) by Charles Hamilton (@MrUn1k0d3r) automates searches and web categorization checking with expireddomains.net and BlueCoat. It can be modified to apply more filters to searches or even perform long term monitoring of assets you register.

Another tool, [DomainHunter](#) by Joe Vest (@joevest) & Andrew Chiles (@andrewchiles), returns BlueCoat/WebPulse, IBM X-Force, and Cisco Talos categorization, domain age, alternate available TLDs, Archive.org links, and an HTML report. Additionally, it performs checks for use in known malware and phishing campaigns using Malwaredomains.com and



MXToolBox. This tool also includes OCR support for bypassing the BlueCoat/WebPulse captchas. Check out the [blog post](#) about the tool's initial release for more details.

Yet another tool, [AIRMASTER](#) by [Max Harley \(@Max\\_68\)](#) uses expireddomains.net and Bluecoat to find categorized domains. This tool uses OCR to bypass the BlueCoat captcha, increasing the search speed.

If a previously-registered domain isn't available or you would prefer a self-registered domain, it's possible to categorize domains yourself. Using the direct links below or a tool like [Chameleon](#) by Dominic Chell ([@domchell](#)). Most categorization products will overlook redirects or cloned content when determining the domain's categorization. For more information about Chameleon usage, check out Dominic's post [Categorisation is not a security boundary](#).

Finally, make sure your DNS settings have propagated correctly.

- [DNS Propagation Checker](#)

## Categorization and Blacklist Checking Resources

---

- [McAfee](#)
- [Fortiguard](#)
- [Symantec + BlueCoat](#)
- [Checkpoint](#) (requires free account)
- [Palo Alto](#)
- [Sophos](#) (submission only; no checking) - Click Submit a Sample -> Web Address
- [TrendMicro](#)
- [Brightcloud](#)
- [Websense](#) (Forcepoint)
- [Lightspeed Systems](#)



- [Chameleon](#)
- [SenderBase](#)
- [MultiBL](#)
- [MXToolBox - Blacklists](#)


# Phishing Setup

## Easy Web-Based Phishing

The words easy and phishing never really seem to go together. Setting up a proper phishing infrastructure can be a real pain. The following tutorial will provide you with the knowledge and tools to quickly setup a phishing server that passes "most" spam filters to-date and provides you with a RoundCube interface for an easy phishing experience including two-way communications with your target. There are many setup's and posts out there regarding phishing. This is just one method.

Once you have a domain that passes the proper checks listed in the previous section and have your phishing server spun-up, you'll need to create a couple "A" records for your domain as pictured.

<input type="checkbox"/>	A Record	@	104.238.144.29	5 min	
<input type="checkbox"/>	A Record	mail	104.238.144.29	5 min	

 ADD NEW RECORD

Next, ssh into your phishing server and make sure you have a proper FQDN hostname listed in your /etc/hosts. Example "127.0.0.1 email.yourphishingserver.com email localhost"

Now, you're going to install the web front-end to phish from in just a few easy steps. Start by downloading the latest "BETA" version of [iRedMail](#) onto your phishing server. Easy way is to right click the download button, copy the link address, use `wget` to download directly onto your phishing server. Next, untar it "`tar -xvf iRedMail-0.9.8-beta2.tar.bz2`". Navigate into the unpacked folder and make the `iRedMail.sh` script executable (`chmod +x iRedMail.sh`). Execute the script as root, follow the prompts, and you'll need to reboot to finish everything.

You'll want to make sure you have all the proper DNS records pointing to your mail server.

(<https://docs.iredmail.org/setup.dns.html>). For DKIM, the new command should be "`amavisd-new showkeys`" to list your DKIM key.

For DMARC we can use (<https://www.unlocktheinbox.com/dmarcwizard/>) to generate our dmarc entry.

**Upgrade to Pro edition to get these features**

- self-service: user is able to login to iRedAdmin-Pro to
  - change password
  - manage mail forwarding
  - manage white/blacklist
  - manage quarantined mails
  - manage spam policy
- Unlimited number of mail lists/aliases
- Per-user real-time quota usage report
- Per-user forwarding, bcc, relay, aliases support
- Alias domain support
- Per-domain bcc, relay, catch-all, account limit control
- Per-domain and per-user service restrictions
- Greylisting control
- Throttling control
- Manage quarantined mails
- View log of received and sent emails
- And many more ...

[Pricing and Upgrade Now](#)

The above tasks can be done in the open source version with e.g. command line tools or phpMyAdmin, etc. but this is more error prone and less easier to do than with the Pro version.

PDFCROWD

**iRedAdmin**  
iRedMail Admin Panel

[Dashboard](#) [Domains and Accounts](#) [Admins](#) [System](#) [+ Add...](#)

### Add user under domain: bypass-bromium.online

**Mail Address \***

noreply @ bypass-bromium.online ▼

**New password \***

.....

**Confirm new password \***

.....

**Display Name**

noreply

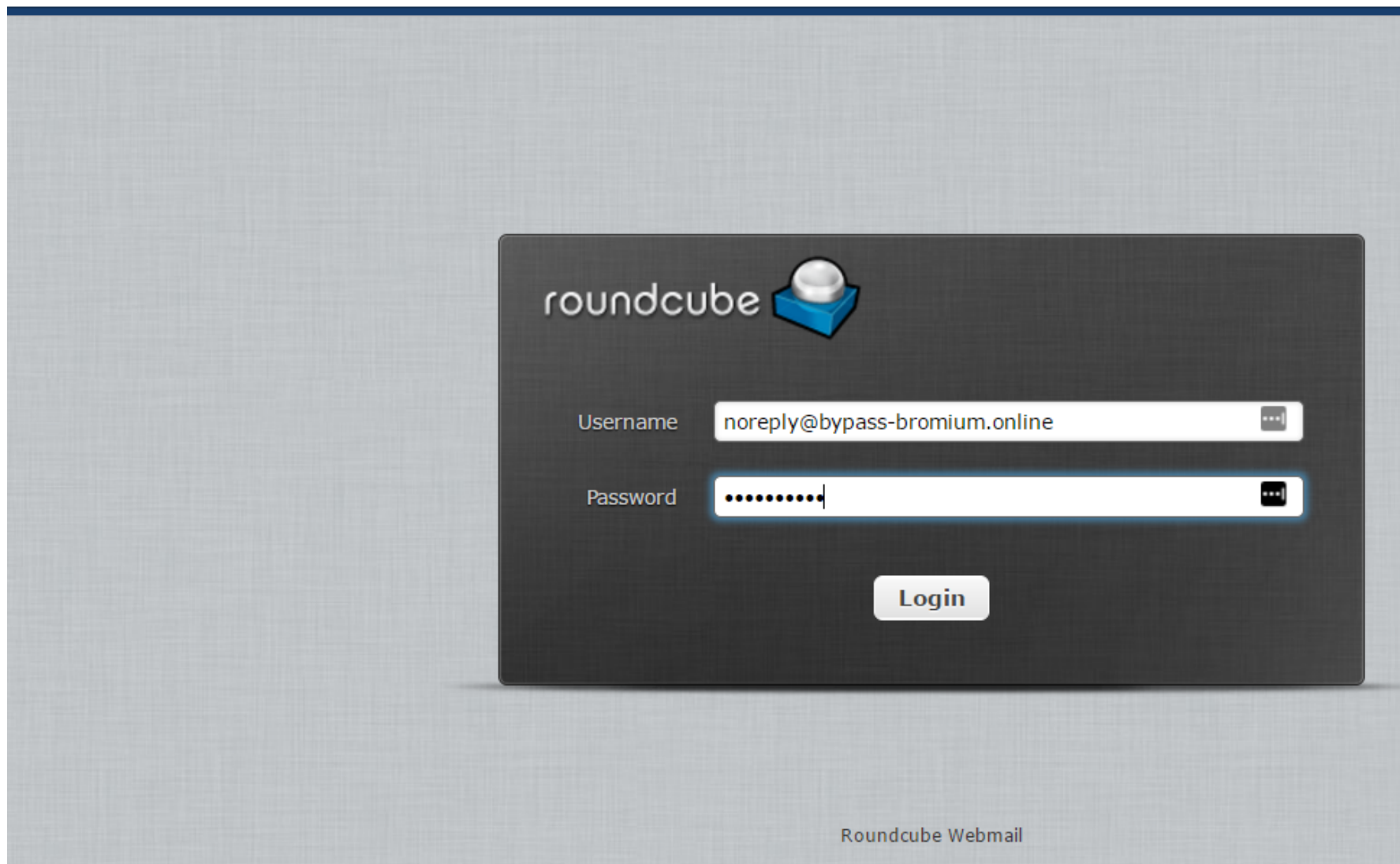
**Mailbox Quota**

100 MB

Add

Login to the RoundCube interface with your new user and phish responsibly!

ps://bypass-bromium.online/mail/?\_task=login



The image shows a web browser window displaying the Roundcube Webmail login page. The URL in the address bar is `ps://bypass-bromium.online/mail/?_task=login`. The login form is centered on a light gray background. It features the Roundcube logo (a blue cube with a white sphere) and the text "roundcube". Below the logo, there are two input fields: "Username" and "Password". The "Username" field contains the text `noreply@bypass-bromium.online`. The "Password" field contains a series of dots, indicating a masked password. Below the input fields is a "Login" button. At the bottom of the page, the text "Roundcube Webmail" is visible.

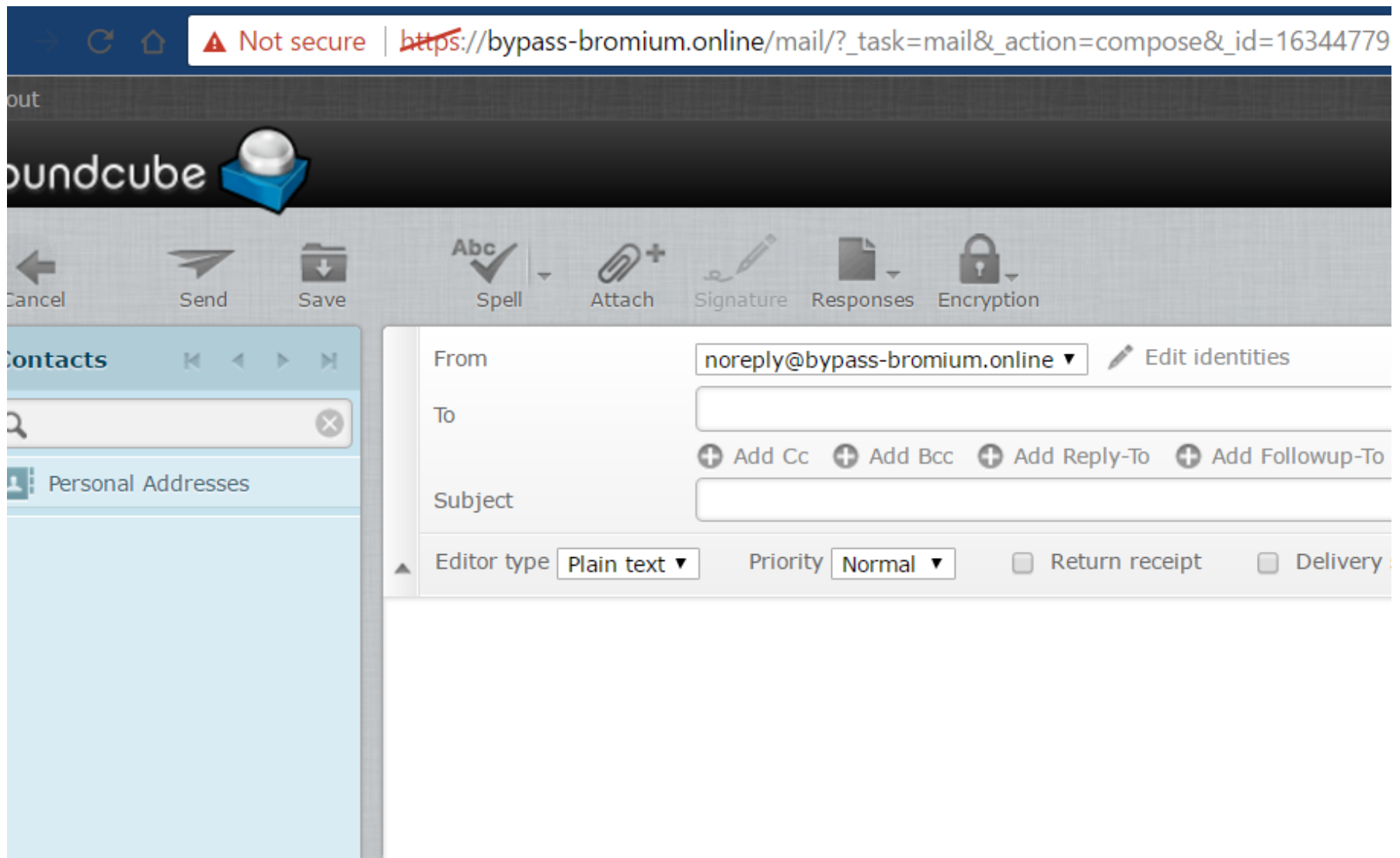
roundcube

Username `noreply@bypass-bromium.online`

Password `.....`

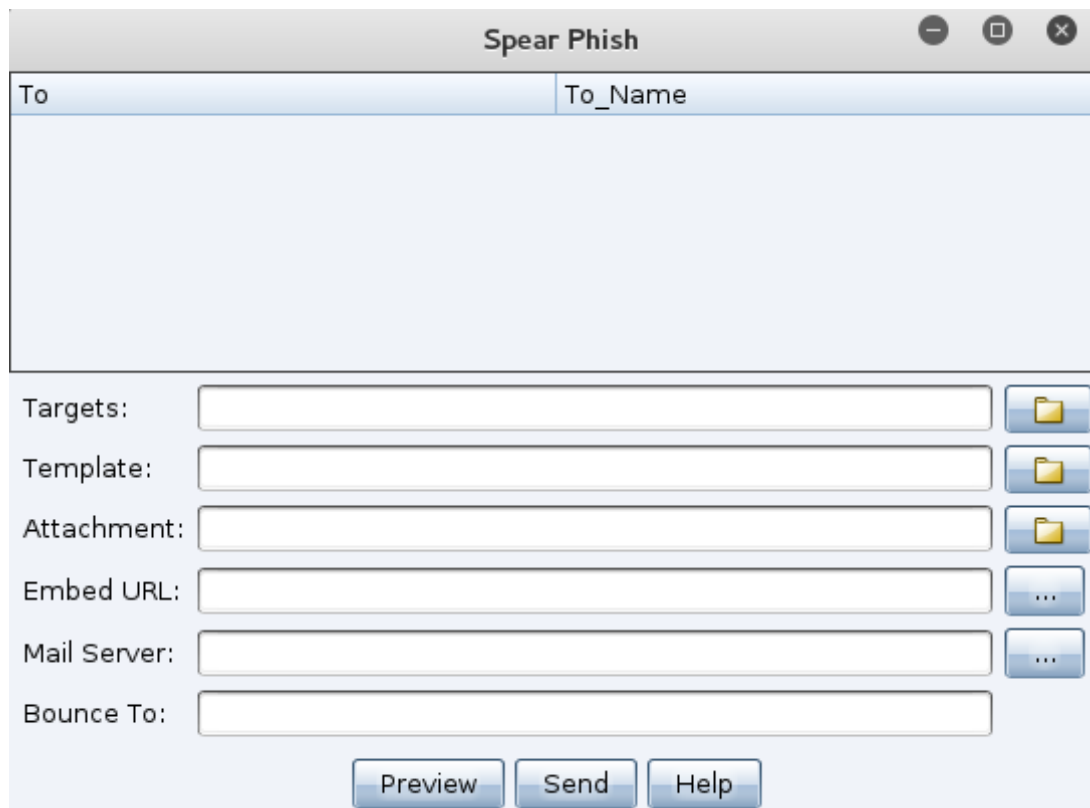
Login

Roundcube Webmail




## Cobalt Strike Phishing


Cobalt Strike provides customizable spearphishing functionality to support pentest or red team email phishing. It supports templates in HTML and/or plaintext formats, attachments, a bounceback address, URL embedding, remote SMTP server usage, and per-message send delays. Another interesting feature is the ability to add a unique token to each user's embedded URL for click tracking.





Spear Phish


To To\_Name

Targets:  

Template:  

Attachment:  

Embed URL:  

Mail Server:  

Bounce To:

For more detailed information, check out these resources:

- [Cobalt Strike - Spear Phishing documentation](#)
- [Cobalt Strike Blog - What's the go-to phishing technique or exploit?](#)
- [Spear phishing with Cobalt Strike - Raphael Mudge](#)



- [Advanced Threat Tactics \(3 of 9\) - Targeted Attacks - Raphael Mudge](#)

## Phishing Frameworks

---

Beyond rolling your own phishing setup or using a pentest or red teaming framework, like Cobalt Strike, there are numerous tools and frameworks dedicated to email phishing. While this wiki won't go into detail about each framework, a few resources for each are collected below:

### Gophish

- [Gophish Official Site](#)
- [Gophish GitHub Repo](#)
- [Gophish User Guide](#)

### Phishing Frenzy

- [Phishing Frenzy Official Site](#)
- [Phishing Frenzy GitHub Repo](#)
- [Introducing Phishing Frenzy - Brandon McCann \(@zeknox\)](#)

### The Social-Engineer Toolkit

- [The Social-Engineer Toolkit GitHub Repo](#)
- [The Social-Engineer Toolkit User Manual](#)

### FiercePhish (formerly FirePhish)

- [FiercePhish GitHub Repo](#)

- [FiercePhish Wiki](#)

# Redirectors

## SMTP

“Redirector” may not be the best word to describe what we’re going to accomplish, but the goal is the same as with our other redirection. We want to remove any traces of our phishing origination from the final email headers and provide a buffer between the victim and our backend server. Ideally, the SMTP redirector will be quick to setup and easy to decommission.

There are two key actions we want to configure an SMTP redirector to perform:

### Sendmail

#### Remove previous server headers

Add the following line to the end of `/etc/mail/sendmail.mc`:

```
define(`confRECEIVED_HEADER', `by $j ($v/$Z)$?r with $r$. id $i; $b')dnl
```

Add to the end of `/etc/mail/access`:

```
IP-to-Team-Server *TAB* RELAY  
Phish-Domain *TAB* RELAY
```

#### [Removing Sender’s IP Address From Email’s Received From Header](#)

## Removing Headers from Postfix setup

### Configure a catch-all address

This will relay any email received to `*@phishdomain.com` to a chosen email address. This is highly useful to receive any responses or bounce-backs to a phishing email.

```
echo PHISH-DOMAIN >> /etc/mail/local-host-names
```

Add the following line right before `//Mailer Definitions//` (towards the end) of `/etc/mail/sendmail.mc`:

```
FEATURE(`virtusertable', `hash -o /etc/mail/virtusertable.db')dnl
```

Add the following line to the end of `/etc/mail/virtusertable`:

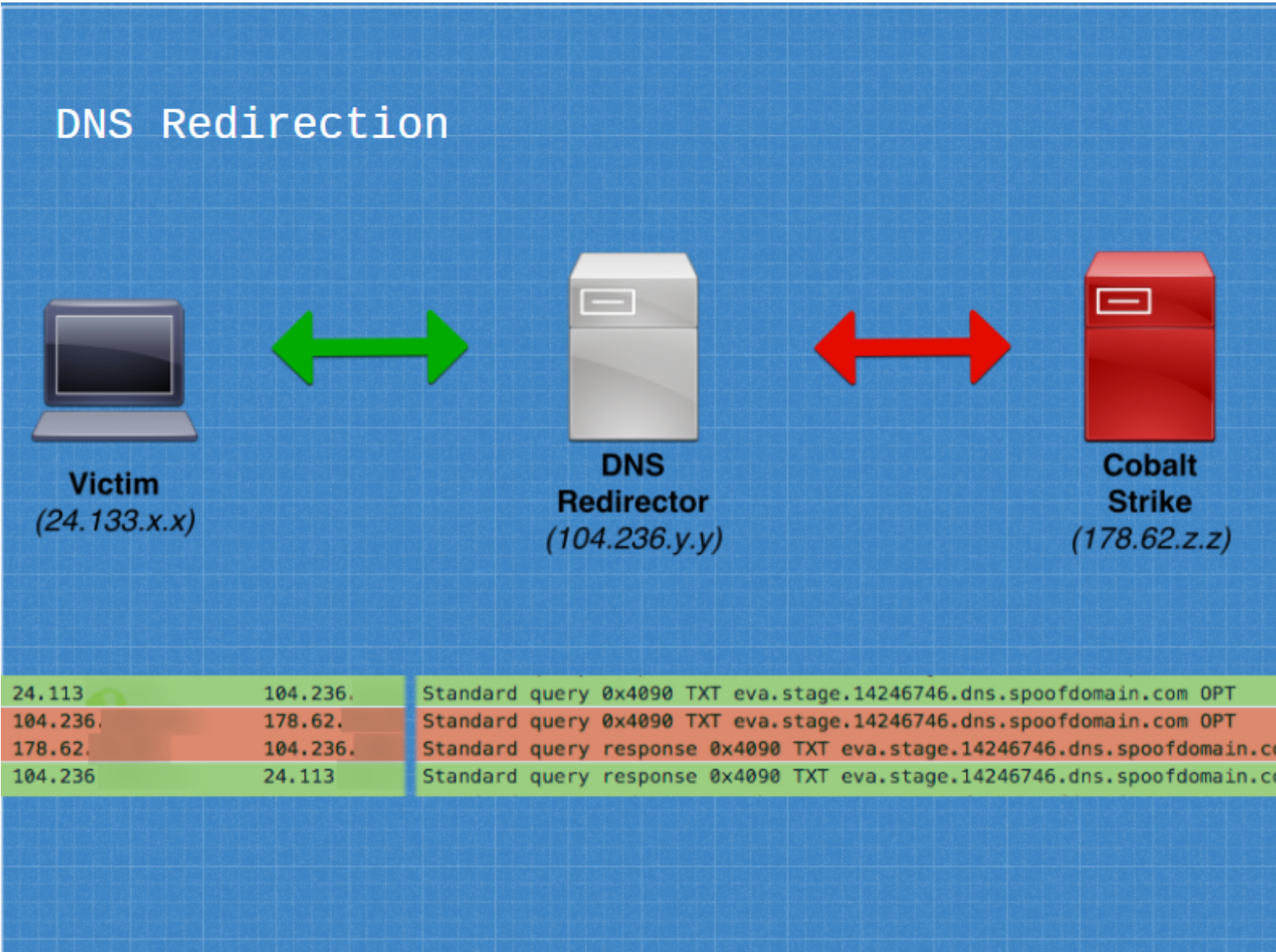
```
@phishdomain.com    external-relay-address
```

*Note: The two fields should be tab-separated*

## Postfix

Postfix provides an easier alternative to sendmail with wider compatibility. Postfix also offers full IMAP support with Dovecot. This allows testers to correspond in real-time with phishing targets who respond to the original message, rather than relying on the catch-all address and having to create a new message using your phishing tool.

A full guide to setting up a Postfix mail server for phishing is available in Julian Catrambone's ([@n0pe\\_sled](#)) post [Mail Servers Made Easy](#).



Note: When using C2 redirectors, a foreign listener should be configured on your post-exploitation framework to send staging traffic through the redirector domain. This will cause the compromised host to stage through the redirector like the C2 traffic itself.

## socat for DNS

socat can be used to redirect incoming DNS packets on port 53 to our team server. While this method works, some user's have reported staging issues with Cobalt Strike and or latency issues using this method. Edit 4/21/2017: The following socat command seems to work well thanks to testing from @xorrior:

```
socat udp4-recvfrom:53,reuseaddr,fork udp4-sendto:<IPADDRESS>; echo -ne
```

[Redirecting Cobalt Strike DNS Beacons - Steve Borosh](#)

## iptables for DNS

iptables DNS forwarding rules have been found to work well with Cobalt Strike. There does not seem to be any of the issues that socat has handling this type of traffic.

An example DNS redirector rule-set is below.

```
iptables -I INPUT -p udp -m udp --dport 53 -j ACCEPT
iptables -t nat -A PREROUTING -p udp --dport 53 -j DNAT --to-destination <IP-GOES-HERE>:53
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -I FORWARD -j ACCEPT
iptables -P FORWARD ACCEPT
sysctl net.ipv4.ip_forward=1
```

Also, change "FORWARD" chain policy to "ACCEPT"

## DNS redirection can also be done behind NAT

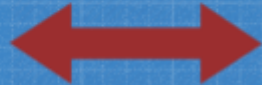
Some may have the requirement or need to host a c2 server on an internal network. Using a combination of IPTABLES, SOCAT, and reverse ssh tunnels, we can certainly achieve this in the following manner.



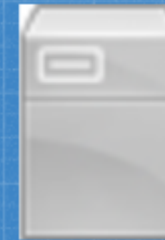
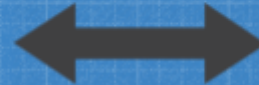
## NAT'd DNS Redirection



**Cobalt Strike**  
(192.168.20.10)  
SOCAT & SSH



**Main Redirector**  
(104.236.x.x)  
SOCAT



**Volatile Redirector**  
(45.63.y.y)  
IPTables

```
# Cobalt Strike Server
socat -t0 -T0 tcp-listen:6667,reuseaddr,fork UDP:localhost:53
ssh user@main_redirector -R 6667:localhost:6667

#Main main_redirector
socat -t0 -T0 udp4-recvfrom:53,reuseaddr,fork tcp:localhost:6667
```

<https://gist.github.com/pcting/1041387>

In this scenario we have our volatile redirector using IPTables to forward all DNS traffic using the rule example described earlier in this section. Next, we create an SSH reverse port forward tunnel from our internal c2 server, to our main redirector.

This will forward any traffic the main redirector receives on port 6667 to the internal c2 server on port 6667. Now, start socat on our team server to fork any of the incoming TCP traffic on port 6667 to UDP port 53 which, is what our DNS c2 needs to listen on. Finally, we similarly setup a socat instance on the main redirector to redirect any incoming UDP port 53 traffic into our SSH tunnel on port 6667.

## HTTP(S)

---

Note: When using C2 redirectors, a foreign listener should be configured on your post-exploitation framework to send staging traffic through the redirector domain. This will cause the compromised host to stage through the redirector like the C2 traffic itself.

### socat vs mod\_rewrite

socat provides a 'dumb pipe' redirection. Any request socat receives on the specified source interface/port is redirected to the destination IP/port. There is no filtering or conditional redirecting. Apache mod\_rewrite, on the other hand, provides a number of methods to strengthen your phishing and increase the resilience of your testing infrastructure. mod\_rewrite has the ability to perform conditional redirection based on request attributes, such as URI, user agent, query string, operating system, and IP. Apache mod\_rewrite uses htaccess files to configure rulesets for how Apache should handle each incoming request. Using these rules, you could, for instance, redirect requests to your server with the default wget user agent to a legitimate page on your target's website.

In short, if your redirector needs to perform conditional redirection or advanced filtering, use Apache mod\_rewrite. Otherwise, socat redirection with optional iptables filtering will suffice.

### socat for HTTP

socat can be used to redirect any incoming TCP packets on a specified port to our team server.

The basic syntax to redirect TCP port 80 on localhost to port 80 on another host is:



```
socat TCP4-LISTEN:80,fork TCP4:<REMOTE-HOST-IP-ADDRESS>:80
```

If your redirector is configured with more than one network interface, socat can be bound to a specific interface, by IP address, with the following syntax:

```
socat TCP4-LISTEN:80,bind=10.0.0.2,fork TCP4:1.2.3.4:80
```

In this example, 10.0.0.2 is one of the redirector's local IP addresses and 1.2.3.4 is the remote team server's IP address.

## iptables for HTTP

In addition to socat, iptables can perform 'dumb pipe' redirection via NAT. To forward the redirector's local port 80 to a remote host, use the following syntax:

```
iptables -I INPUT -p tcp -m tcp --dport 80 -j ACCEPT
iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination <REMOTE-HOST-IP-ADDRESS>:80
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -I FORWARD -j ACCEPT
iptables -P FORWARD ACCEPT
sysctl net.ipv4.ip_forward=1
```

## SSH for HTTP

We have previously covered using SSH for DNS tunnels. SSH works as a solid, and robust means to break through NAT and obtain a way for the implant to connect to a redirector and into your server environment. First you must set up GatewayPorts forwarding or it won't work, using the following syntax on the redirector:

```
nano /etc/ssh/sshd_config add GatewayPorts yes
```

To forward the redirector's local port 80 to your internal teamsrver, use the following syntax on the internal server:

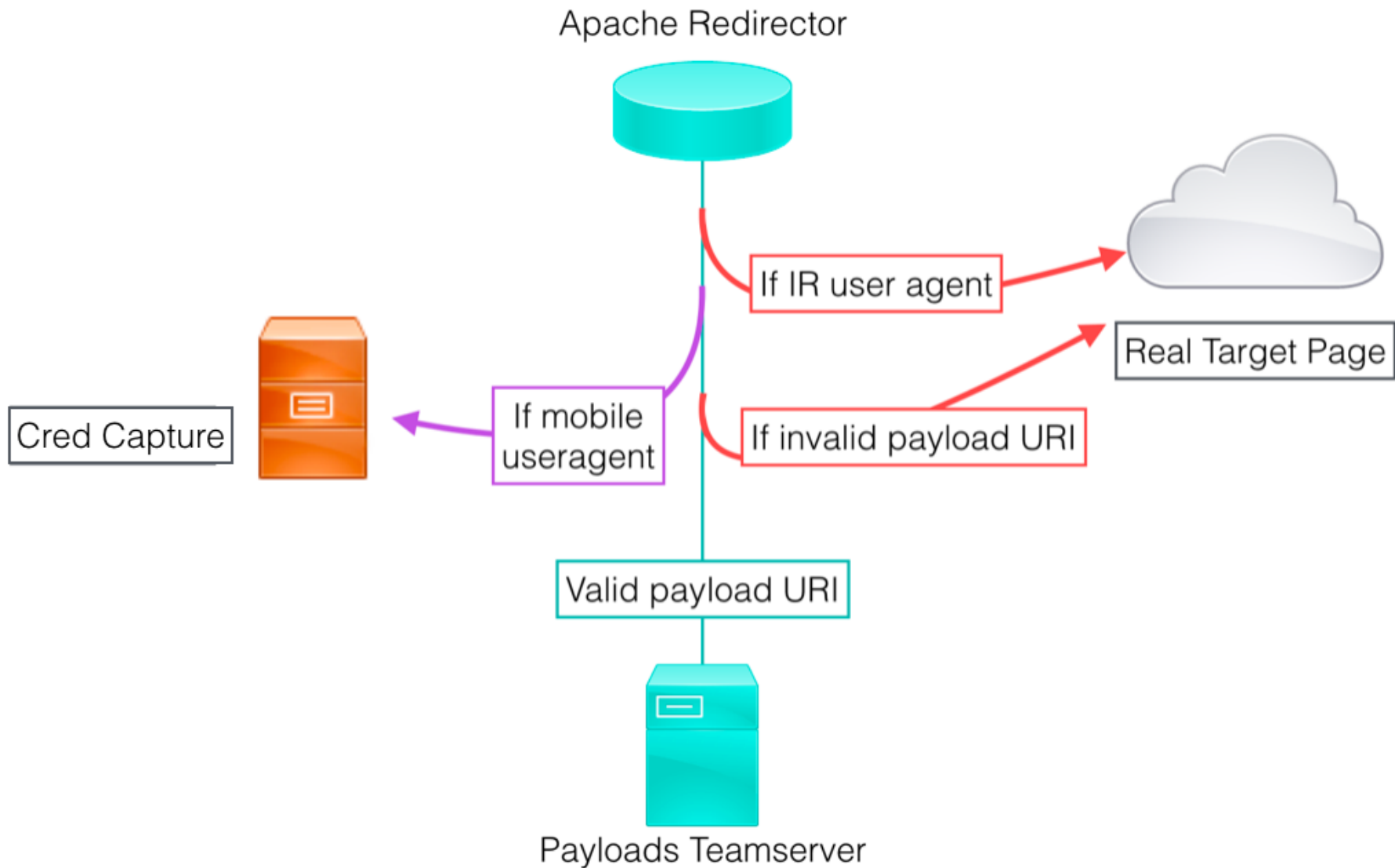
```
tmux new -S redir80  
ssh <redirector> -R *:80:localhost:80  
Ctrl+B, D
```

You can also forward more than one port, for example if you want 443 and 80 to be open all at once:

```
tmux new -S redir80443  
ssh <redirector> -R *:80:localhost:80 -R *:443:localhost:443  
Ctrl+B, D
```

## Payloads and Web Redirection

When serving payload and web resources, we want to minimize the ability for incident responders to review files and increase the chances of successfully executing the payload, whether to establish C2 or gather intelligence.



Apache Mod\_Rewrite usage and examples by Jeff Dimmock:

- [Strengthen Your Phishing with Apache mod\\_rewrite](#)
- [Invalid URI Redirection with Apache mod\\_rewrite](#)

- [Operating System Based Redirection with Apache mod\\_rewrite](#)
- [Combatting Incident Responders with Apache mod\\_rewrite](#)
- [Expire Phishing Links with Apache RewriteMap](#)
- [Apache mod\\_rewrite Grab Bag](#)
- [Serving Random Payloads with Apache mod\\_rewrite](#)

Other Apache mod\_rewrite usage and examples:

- [mod\\_rewrite rule to evade vendor sandboxes from Jason Lang @curi0usjack](#)
- [Serving random payloads with NGINX - Gist by jivoi](#)

To automatically set up Apache Mod\_Rewrite on a redirector server, check out Julain Catrambone's ([@n0pe\\_sled](#)) blog post [Mod\\_Rewrite Automatic Setup](#) and the [accompanying tool](#).

## C2 Redirection

The intention behind redirecting C2 traffic is twofold: obscure the backend team server and appear to be a legitimate website if browsed to by an incident responder. Through the use of Apache mod\_rewrite and [customized C2 profiles](#) or other proxying (such as with Flask), we can reliably filter the real C2 traffic from investigative traffic.

- [Cobalt Strike HTTP C2 Redirectors with Apache mod\\_rewrite - Jeff Dimmock](#)
- [Securing your Empire C2 with Apache mod\\_rewrite - Gabriel Mathenge \(@\\_theVIVI\)](#)
- [Expand Your Horizon Red Team – Modern SAAS C2 - Alex Rymdeko-Harvey \(@killswitch-gui\)](#)
- [Hybrid Cobalt Strike Redirectors - Zach Grace \(@ztgrace\) and @m0ther\\_](#)

## C2 Redirection with HTTPS

Building on "C2 Redirection" above, another method is to have your redirecting server use Apache's SSL Proxy Engine to accept inbound SSL requests, and proxy those to requests to a reverse-HTTPS listener. Encryption is used at all stages, and you can rotate SSL certificates on your redirector as needed.

To make this work with your `mod_rewrite` rules, you need to place your rules in **`"/etc/apache2/sites-available/000-default-le-ssl.conf"`** assuming you've used LetsEncrypt (aka CertBot) to install your certificate. Also, to enable the SSL ProxyPass engine, you'll need the following lines in that same config file:

```
# Enable the Proxy Engine
SSLProxyEngine On

# Tell the Proxy Engine where to forward your requests
ProxyPass / https://DESTINATION_C2_URL:443/
ProxyPassReverse / https://DESTINATION_C2_URL:443/

# Disable Cert checking, useful if you're using a self-signed cert
SSLProxyCheckPeerCN off
SSLProxyCheckPeerName off
SSLProxyCheckPeerExpire off
```

## Other Apache `mod_rewrite` Resources

- [Automating Apache `mod\_rewrite` and Cobalt Strike Profiles](#)
- [mod\\_rewrite-cheatsheet.com](#)
- [Official Apache 2.4 `mod\_rewrite` Documentation](#)
- [Apache `mod\_rewrite` Introduction](#)
- [An In-Depth Guide to `mod\_rewrite` for Apache](#)
- [Mod\\_Rewrite/.htaccess Syntax Checker](#)

# Modifying C2 Traffic

---

## Cobalt Strike

---

Cobalt Strike modifies its traffic with Malleable C2 profiles. Profiles provide highly-customizable options for modifying how your server's C2 traffic will look on the wire. Malleable C2 profiles can be used to strengthen incident response evasion, impersonate known adversaries, or masquerade as legitimate internal applications used by the target.

- [Malleable C2 Profiles - GitHub](#)
- [Malleable Command and Control Documentation - cobaltstrike.com](#)
- [Cobalt Strike 2.0 - Malleable Command and Control - Raphael Mudge](#)
- [Cobalt Strike 3.6 - A Path for Privilege Escalation - Raphael Mudge](#)
- [A Brave New World: Malleable C2 - Will Schroeder \(@harmj0y\)](#)
- [How to Write Malleable C2 Profiles for Cobalt Strike - Jeff Dimmock](#)

## Empire

---

Empire uses Communication Profiles, which provide customization options for the GET request URIs, user agent, and headers. The profile consists of each element, separated by the pipe character, and set with the `set DefaultProfile` option in the `listeners` context menu.

Here is a sample default profile:

```
"/CWoNaJLBo/VTNeww11212/|Mozilla/4.0 (compatible; MSIE 6.0;Windows NT 5.1)|Accept:image/gif, image/x-xbitma
```

Alternatively, the DefaultProfile value can be set by modifying the file `/setup/setup_database.py` before Empire's initial setup. This will change the default Communication Profile that Empire will use.

In addition to the Communication Profile, consider customizing the Empire server's staging URIs, server headers, and default webpage content by following the steps presented in Joe Vest's (@joevest) post [Empire - Modifying Server C2 Indicators](#).

- [Default Empire Communication Profiles \(in Empire GitHub repo\)](#)
- [How to Make Communication Profiles for Empire - Jeff Dimmock](#)

## Third-Party C2 Channels

---

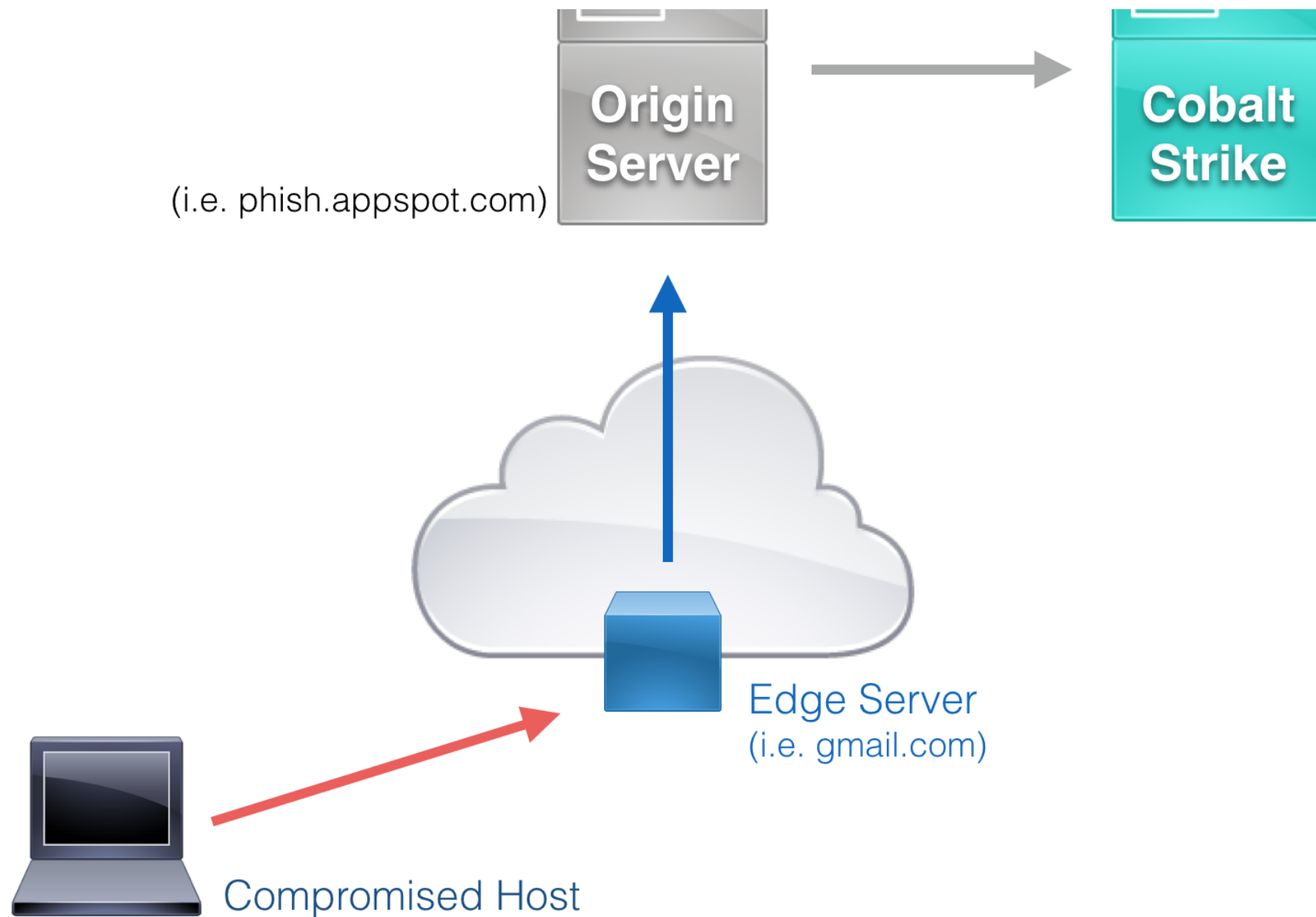
Leveraging trusted, legitimate web services for C2 can provide a valuable leg-up over using domains and infrastructure you've configured yourself. Configuration time and complexity varies based on the technique and service being used. A popular example of leveraging third-party services for C2 redirection is Domain Fronting.

### Domain Fronting

---

Domain Fronting is a technique used by censorship evasion services and apps to route traffic through legitimate and highly-trusted domains. Popular services that support Domain Fronting include [Google App Engine](#), [Amazon CloudFront](#), and [Microsoft Azure](#). In a nutshell, traffic uses the DNS and SNI name of the trusted service provider, Google is used in the example below. When the traffic is received by the Edge Server (ex: located at gmail.com), the packet is forwarded to the Origin Server (ex: phish.appspot.com) specified in the packet's Host header. Depending on the service provider, the Origin Server will either directly forward traffic to a specified domain, which we'll point to our team server, or a proxy app will be required to perform the final hop forwarding.





For more detailed information about how Domain Fronting works, see the whitepaper [Blocking-resistant communication through domain fronting](#) and the TOR Project's [meek documentation](#)



In addition to the standard frontable domains, such as any google.com domain, it's possible to leverage other legitimate domains for fronting.

For more information about hunting frontable domains, check out:

- [Domain Fronting via Cloudfront Alternate Domains - Vincent Yiu \(@vysecurity\)](#)
- [Finding Domain frontable Azure domains - thoth / Fionnbharr \(@a\\_profligate\)](#)
- [Google Groups: Blog post on finding 2000+ Azure domains using Censys](#)
- [FindFrontableDomains tool - Steve Borosh \(@rvrsh3ll\)](#)

## Further Resources on Domain Fronting

- [Simplifying Domain Fronting - Tim Malcomvetter \(@malcomvetter\)](#)
- [High-reputation Redirectors and Domain Fronting - Raphael Mudge](#)
- [Empire Domain Fronting - Chris Ross \(@xorrior\)](#)
- [Escape and Evasion Egressing Restricted Networks - Tom Steele \(@\\_tomsteele\) and Chris Patten](#)
- [Red Team Insights on HTTPS Domain Fronting Google Hosts Using Cobalt Strike - Will Vandevanter and Shay Nahari of CyberArk](#)
- [SSL Domain Fronting 101 - Steve Borosh \(@424f424f\)](#)
- [How I Identified 93k Domain-Frontable CloudFront Domains - Chris Myers \(@SWIZZLEZ\\_\) and Barrett Adams \(@PEEWPW\)](#)
- [Domain Fronting: Who Am I? - Vincent Yiu \(@vysecurity\)](#)
- [Validated CloudFront SSL Domains - Vincent Yiu \(@vysecurity\)](#)
- [CloudFront Hijacking - Matt Westfall \(@disloops\)](#)
- [CloudFront GitHub Repo - MindPointGroup](#)
- [Metasploit Domain Fronting With Microsoft Azure \(@ch1gg1ns\)](#)

## PaaS Redirectors

---

Many PaaS and SaaS providers provide a static subdomain or URL for use with a provisioned instance. If the associated domain is generally highly trusted, the instances could provide extra trust to your C2 infrastructure over a purchased domain and VPS.

To set the redirection up, you will need to identify a service that issues a static subdomain or URL as part of an instance. Then, either the instance will need to be configured with network or application-based redirection. The instance will act as a proxy, similar to the other redirectors discussed on this wiki.

Specific implementation can vary greatly based on the service; however, for an example using Heroku, check out the blog post [Expand Your Horizon Red Team – Modern SaaS C2](#) by [Alex Rymdeko-Harvey \(@Killswitch\\_GUI\)](#).

Another interesting technique that merits further research is the use of overly-permissive Amazon S3 buckets for C2. Check out the post [S3 Buckets for Good and Evil](#) by [Andrew Luke \(@Sw4mp\\_f0x\)](#) for more details on how S3 buckets could be used for C2. This technique could be combined with the third-party C2 capabilities of Empire to use the target's legitimate S3 buckets against them.

## Other Third-Party C2

---

Other third-party services have been used in the wild for C2 in the past. Leveraging third-party websites that allow for the rapid posting or modification of user-generated content can help you evade reputation-based controls, especially if the third-party site is generally trusted.

Check out these resources for other third-party C2 options:

- [A stealthy Python based Windows backdoor that uses Github as a C&C server](#) - maldevel at securityblog.gr
- [External C2 \(Third-Party Command and Control\)](#) - Cobalt Strike Documentation
- [Cobalt Strike over external C2 – beacon home in the most obscure ways](#) - Mark Bergman at outflank.nl

- [“Tasking” Office 365 for Cobalt Strike C2](#) - William Knowles (@william\_knows)
- [External C2 for Cobalt Strike](#) - Ryan Hanson (@ryhanson)
- [External C2 framework for Cobalt Strike](#) - Jonathan Echavarria (@Und3rf10w)
- [External C2 framework \(GitHub Repo\)](#) - Jonathan Echavarria (@Und3rf10w)
- [Hiding in the Cloud: Cobalt Strike Beacon C2 using Amazon APIs](#) - Rhino Security Labs
- [Exploring Cobalt Strike's ExternalC2 framework](#) - Adam (@xpn)

## Obscuring Infrastructure

---

Attack infrastructure is often easy to identify, appearing like a shell of a legitimate server. We will need to take additional steps with our infrastructure to increase the likelihood of blending in with real servers amongst either the target organization or services the target may conceivably use.

[Redirectors](#) can help blend in by [redirecting invalid URIs](#), [expiring phishing payload links](#), or [blocking common incident responder techniques](#); however, attention should also be paid to the underlying host and its indicators.

For example, in the post [Fall of an Empire](#), John Menerick (@Lord\_SQL) covers methods to detect Empire servers on the internet.

To combat these and similar indicators, it's a good idea to [modify C2 traffic patterns](#), modify server landing pages, restrict open ports, and modify default response headers.

For more details about how to do these and other tactics for multiple attack frameworks, check out these posts:

- [Empire – Modifying Server C2 Indicators](#) - Andrew Chiles
- [Hunting Red Team Empire C2 Infrastructure](#) - chokepoint.net
- [Hunting Red Team Meterpreter C2 Infrastructure](#) - chokepoint.net

- [Identifying Empire HTTP Listeners \(Tenable Blog\) - Jacob Baines](#)

## Securing Infrastructure

---

Attack infrastructure can be attacked just the same as any other internet-connected host, and it should be considered HIGHLY sensitive due to the data in use and connections into target environments.

In 2016, remote code execution vulnerabilities were disclosed on the most common attack tools:

- [2016 Metasploit RCE Static Key Deserialization](#)
- [2017 Metasploit Meterpreter Dir Traversal Bugs](#)
- [Empire Fails - Will Schroeder](#)
- [Cobalt Strike 3.5.1 Important Security Update - Raphael Mudge](#)

**iptables** should be used to filter unwanted traffic and restrict traffic between required infrastructure elements. For example, if a Cobalt Strike team server will only serve assets to an Apache redirector, iptables rules should only allow port 80 from the redirector's source IP. This is especially important for any management interfaces, such as SSH or Cobalt Strike's default port 50050. Also consider blocking non-target country IPs. As an alternative, consider using hypervisor firewalls provided by your VPS providers. For example, Digital Ocean offers [Cloud Firewalls](#) that can protect one or multiple droplets.

**chattr** can be used on team servers to prevent cron directories from being modified. Using chattr, you can restrict any user, including root, from modifying a file until the chattr attribute is removed.

**SSH** should be limited to public-key authentication only and configured to use limited-rights users for initial login. For added security, consider adding multi-factor authentication to SSH.

**Update!** No securing list is complete without a reminder to regularly update systems and apply hot-fixes as needed to remediate vulnerabilities.

Of course, this list is not exhaustive of what you can do to secure a team server. Follow common hardening practices on all infrastructure:

- [Red Hat Enterprise Linux 6 Security Guide](#)
- [Debian Documentation on Hardening](#)
- [Securing Debian Manual](#)
- [20 Linux Server Hardening Security Tips - nixCraft](#)
- [SANS Linux Security Checklists](#)
- [Docker Your Command & Control \(C2\) - Alex Rymdeko-Harvey \(@killswitch\\_gui\)](#)

## Specific Hardening Resources

---

There are a number of resources available online discussing the secure setup and design of infrastructures. Not every design consideration will be appropriate for every attack infrastructure, but it's useful to know what options are available and what other testers are doing.

Here are some of those resources:

- [Responsible Red Teams - Tim MalcomVetter \(@malcomvetter\)](#)
- [Safe Red Team Infrastructure - Tim MalcomVetter \(@malcomvetter\)](#)
- [Red Team Infrastructure - AWS Encrypted EBS - @\\_rastamouse](#)

## Automating Deployments

---

The topics covered in this wiki strengthen attack infrastructures, but generally require a good deal of time to design and implement. Automation can be used to greatly reduce deployment times, allowing you to deploy more complex setups in less time.

Check out these resources about attack infrastructure automation:

- [Automated Red Team Infrastructure Deployment with Terraform - Part 1](#) - @\_RastaMouse
- [Automated Red Team Infrastructure Deployment with Terraform - Part 2](#) - @\_RastaMouse
- [Mod\\_Rewrite Automatic Setup](#) - Julian Catrambone (@n0pe\_sled)
- [Automated Empire Infrastructure](#) - Jeremy Johnson (@beyondnegative)
- [RTOps: Automating Redirector Deployment With Ansible](#) - Kevin Dick
- [Automating Gophish Releases With Ansible and Docker](#) - Jordan Wright (@jw\_sec)
- [Red Baron GitHub Repo](#) - Marcello (@byt3bl33d3r)
- [Automating Apache mod\\_rewrite and Cobalt Strike Malleable C2 for Intelligent Redirection](#) - Joe Vest (@joevest)

## General Tips

---

- **Document everything** - Running a complex Red Team infrastructure means many moving parts. Be sure to document each asset's function and where its traffic is sent.
- **Split assets among different service providers and regions** - Infrastructure assets should be spread across multiple service providers and geographic regions. Blue Team members may raise monitoring thresholds against providers identified as actively performing an attack and may even outright block a given service provider. Note: keep international privacy laws in mind if sending encrypted or sensitive data across borders.
- **Don't go overboard** - It's easy to get excited about advanced techniques and want to throw the kitchen sink at a target. If you are emulating a specific adversarial threat, only leverage techniques the real threat actor used or techniques within the skillset of the threat actor. If your red team testing will attack the same target long-term, consider starting "easy" and working through the more advanced tradecraft as your assessments go on. Evolving the red team's technique alongside the blue team's will consistently push the organization forward, whereas hitting the blue team with everything at once may overwhelm the blue team and slow the learning process.

- **Monitor logs** - All logs should be monitored throughout the engagement: SMTP logs, Apache logs, tcpdump on socat redirectors, iptables logs (specific to traffic forwarding or targeted filtering), weblogs, Cobalt Strike/Empire/MSF logs. Forward logs to a central location, such as with [rsyslog](#), for easier monitoring. Operator terminal data retention may come in handy for going over an historical command useage during an operation. @Killswitch\_GUI created an easy-to-use program named ITerm that will log all bash terminal commands to a central location. [Log all terminal output with ITerm](#)
- **Implement high-value event alerting** - Configure the attack infrastructure to generate alerts for high-value events, such as new C2 sessions or credential capture hits. One popular way of implementing alerting is via a chat platform's API, such as Slack. Check out the following posts about Slack alerting: [Slack Shell Bot - Russel Van Tuyl \(@Ne0nd0g\)](#), [Slack Notifications for Cobalt Strike - Andrew Chiles \(@AndrewChiles\)](#), [Slack Bots for Trolls and Work - Jeff Dimmock \(@bluscreenfojeff\)](#)
- **Fingerprint incident response** - If possible, try to passively or actively fingerprint IR actions before the assessment starts. For example, send a mediocre phishing email to the target (using unrelated infrastructure) and monitor traffic that infrastructure receives. IR team investigations can disclose a good deal of information about how the team operates and what infrastructure they use. If this can be determined ahead of the assessment, it can be filtered or redirected outright.

## Thanks to Contributors

---

A BIG THANK YOU to all the following people (listed alphabetically) who contributed tools, tips, or links to include in the wiki, and another THANK YOU to anyone who wrote a tool or post referenced in this wiki!

- [@andrewchiles](#) - Andrew Chiles
- [@armitagehacker](#) - Raphael Mudge
- [@beyondnegative](#) - Jeremy Johnson
- [@bspence7337](#)

- [@domchell](#) - Dominic Chell
- [@jivoi](#) - EK
- [@joevest](#) - Joe Vest
- [@killswitch\\_gui](#) - Alex Rymdeko-Harvey
- [@ne0nd0g](#) - Russel Van Tuyl
- [@n0pe\\_sled](#) - Julian Catrambone
- [@\\_RastaMouse](#)
- [@tifkin\\_](#) - Lee Christensen
- [@Und3rf10w](#) - Jonathan Echavarria
- [@vysecurity](#) - Vincent Yiu
- [@xorrior](#) - Chris Ross

