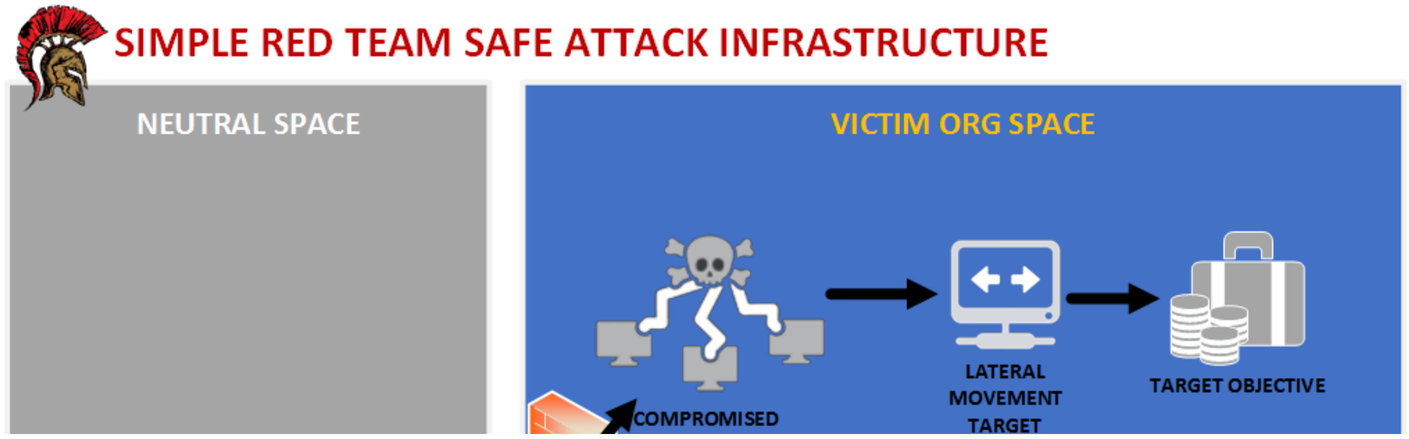
Tim MalcomVetter Follow
Red Team Leader at Fortune 1. I left my clever profile in my other social network: https://www.linkedin.com/in/malcomvetter
Feb 21 · 5 min read

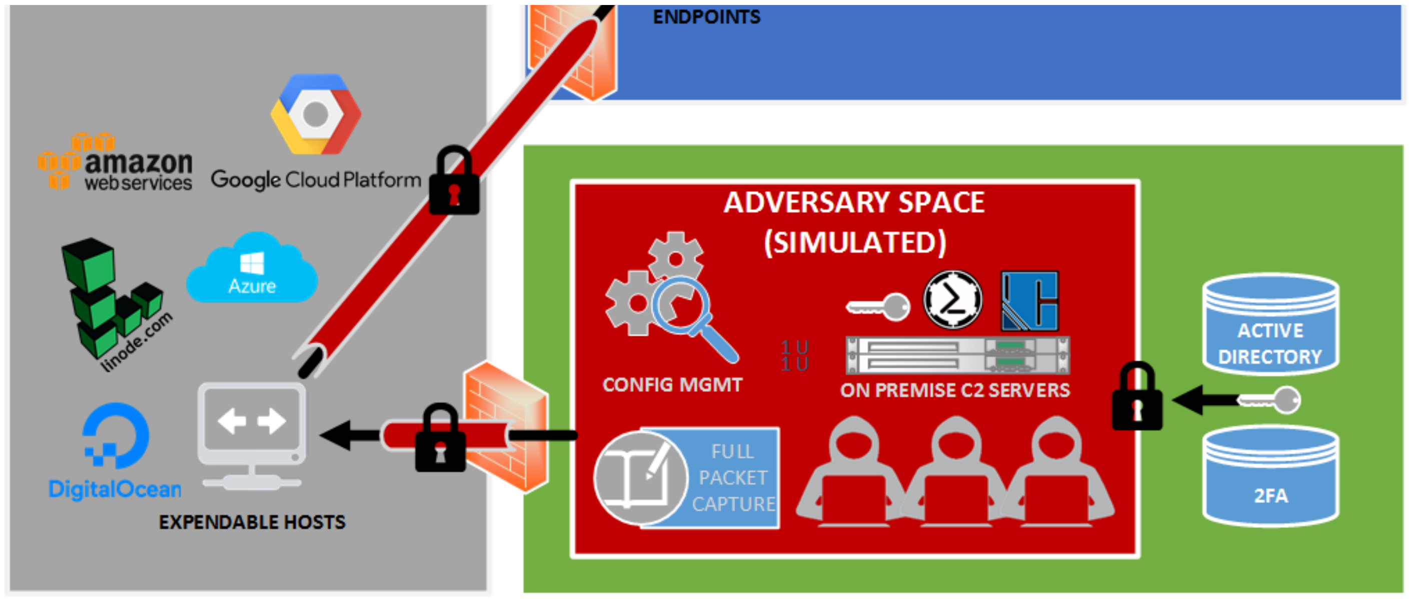
Safe Red Team Infrastructure

This is a quick follow-up to "Responsible Red Teams." This walks at a high-level through creating a safe red team infrastructure that is hosted in your company's protected data center (firewalls, IPS, logging, packet capture, environmentals, door locks, man traps, cameras, locks, armed guards, concrete planters, tank/car bomb traps, violent yard gnomes, what-have-you). What you will be building is the digital equivalent to this analog (I frequently refer to this internal water park tube slide that goes out of the building and back in).

This waterpark slide has been my analog for this Red Team infrastructure model

Note: this is for C2 callbacks only, and expects a similar architecture for phishing, etc.





A simple model that works for consulting or internal corporate Red Teams.

1. Have a hypervisor or physical machines on-premise, without being directly faced to the internet, but having internet connectivity (bonus points, set explicit egress firewall rules for these servers to the host/port that follows below).

2. Install the C2 server of your choosing (Empire and Cobalt Strike are two common examples) on a server in your on-premise data center. We'll assume it lives at 192.168.1.100 (non-routable) and that the C2 server binds to port 443.

3. Procure cloud VMs of the providers of your choice. For good tradecraft, use a variety for your sophistication/emulation goals. In this example, we'll say it lives at 2.2.2.2.

4. Procure DNS domains (e.g. evil.com) that match these goals. Assign the DNS records to point to the cloud VM (2.2.2.2).

5. Have the internal C2 server make an outbound SSH connection through its egress firewalls to the cloud host (2.2.2.2) and establish a REVERSE PORT FORWARD (this seems much more confusing conceptually than it is in implementation). Reverse port forwards bind a local port (e.g. port 2222 on 127.0.0.1 of the cloud VM) to send traffic across the incoming SSH connection to a port defined on the other (calling end, the server in your data center) host (e.g. port 443 on 192.168.1.100). On linux, this can be wrapped with autossh, which is handy at keeping the connection alive.
[Note: this outward directionality of the SSH tunnel is intentional, because it prevents exposing the internal on-premise servers to direct access from Internet hosts. Do not reverse that directionality.]

6. On the Cloud host, create a TCP redirect from the public interface's TCP 443 port to the localhost port 2222 so that traffic rides the SSH tunnel back to your protected on-premise C2 server. On linux, this is a single line call of socat. (Bonus points: socat has parameters to implement ingress filtering by IP range, to keep certain bots or pests off your C2 server, which is useful if you know your victim's IP address space and it matches adversary emulation goals).

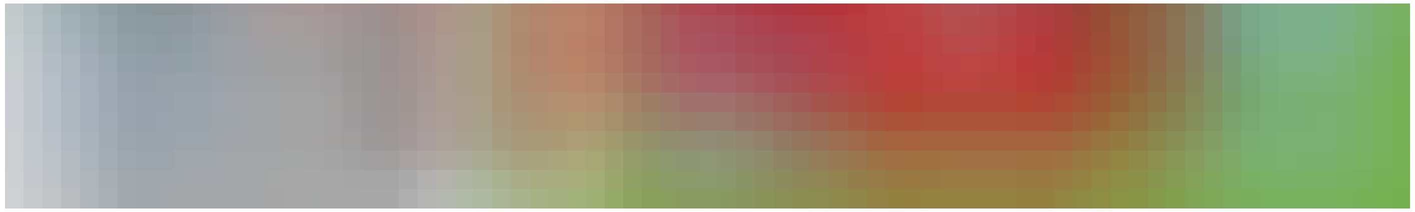
7. On your internal on-premise host, install your SSL certificate, signed by the certificate authority of your choosing. Remember you will use the external DNS (e.g. evil.com) for the common name of the certificate. This way all traffic from victim endpoints will flow through the TCP redirector and be decrypted only on the on-premise server. Never install the certificate/private key on the cloud host or use unencrypted protocols (e.g. HTTP) with this setup or the cloud host provider will have access to your target victim's data.

You can augment this with some advanced features.

Domain Fronting

This same concept can be applied using Domain Fronting (for more info on this, see “Simplifying Domain Fronting”). All we have to do is add in a Content Delivery Network (CDN) that supports domain fronting and locate a frontable DNS domain name (out of scope for this article, I’m assuming you’ve already done that).

This looks like this for a consulting red team:


A more advanced consulting Red Team model with Domain Fronting as an option.

1. Build the above steps in the simple, non-domain fronted model.

2. Point the CDN origin host to the domain name you're using for 2.2.2.2.

3. Verify connectivity to the C2 server using the DNS domain name issued by the CDN service.

4. Swap the issued hostname from the above step into the host header of each request of your payloads, but use the domain name of another CDN customer (a high reputation site you don't control—again this selection is out of scope for this article).

5. Verify connectivity using payloads talking to the high-reputation domain with your CDN domain in the host header of each HTTPS request. This will result in a TLS connection between the CDN and what appears to be your

cloud host, but in actuality will be between the CDN and your on-premise host since the SSL certificates/keys will reside there.

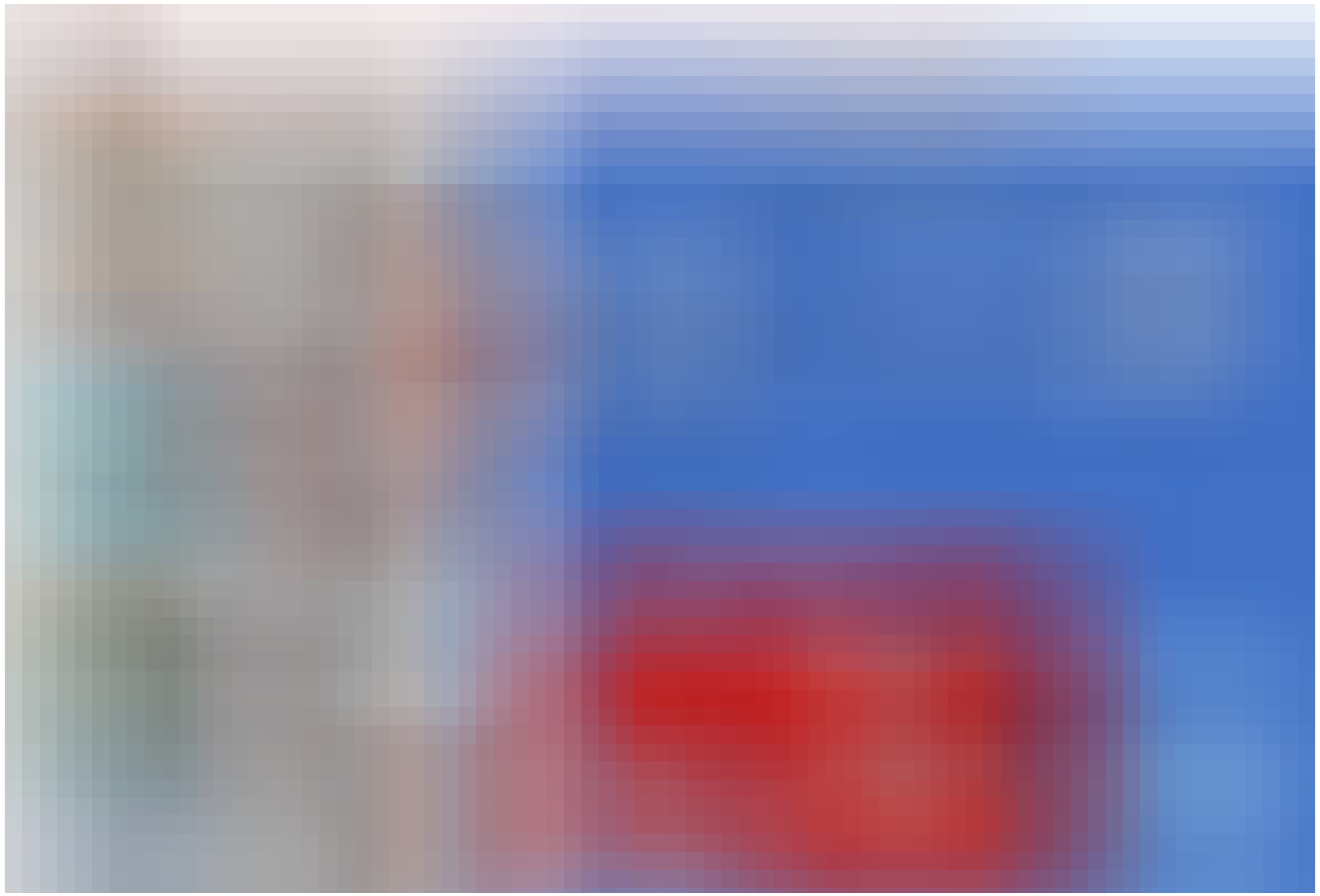
IP Laundry

If you're an internal Red Team and you want to keep your honest Blue Team honest, then you can implement an IP laundering process. The idea is the exact same as the first set of instructions, except you'll chain 2 cloud hosts together. The first host does nothing but operate a series of extra TCP redirector tunnels. No payloads, C2 callbacks, or any other traffic will talk directly to it. It's simply a link in the chain. This way, a blue teamer performing traffic analysis and sneaking a peek at the corporate red team infrastructure can NOT know exactly what is going on. All they'll see is traffic from known red team servers to an IP address on the Internet that is not involved in any traffic to corporate endpoints. The trail is washed.

This architecture also has the side benefit of limiting firewall rule changes to the corporate firewalls, since traffic is only known to be initiated from red team servers to a dedicated host in a cloud provider.

For implementation, repeat steps 4–6 in the first section above, linking the laundry host to the “expendable” host.

This can also be combined with advanced C2 concepts like domain fronting as well, as is pictured below.



An internal corporate Red Team model to keep C2/data on-premise but simulate external threat actors.

Credential Phishes

It’s also possible to replicate this model for credential phishes. Simply install a web application with the phish login form and the necessary code to receive an HTTPS request, parse the credentials, and log them safely to disk, on the C2 server instead of, say, Cobalt Strike. [This can be done in parallel on a second internal host using a second copy of the external infrastructure so that it can be live at the same time as your C2 server.] This will result in credentials being sent end-to-end encrypted using the SSL certificate and private key of your on-premise server without exposing credentials to the cloud host.

Tradecraft Considerations

To maximize tradecraft, a variety of cloud host providers, geographic locations, DNS domains, DNS registrars, domain categories, and C2 server types should be employed. The more variety, the better. Proficient Red Teams often layer this architecture to minimize impact when the Blue Team identifies and blocks traffic to/from a single C2 listener/path.

Be creative.

Red Team Information Security Hacking Penetration Testing

Tim MalcomVetter

Follow

Red Team Leader at Fortune 1. I left my clever profile in my other social network: https://www.linkedin.com/in/malcomvetter

Tim MalcomVetter
3 min read

15

Red Teams Blog
2 min read

6

SPARTaN
8 min read

61

Responses

Write a response…