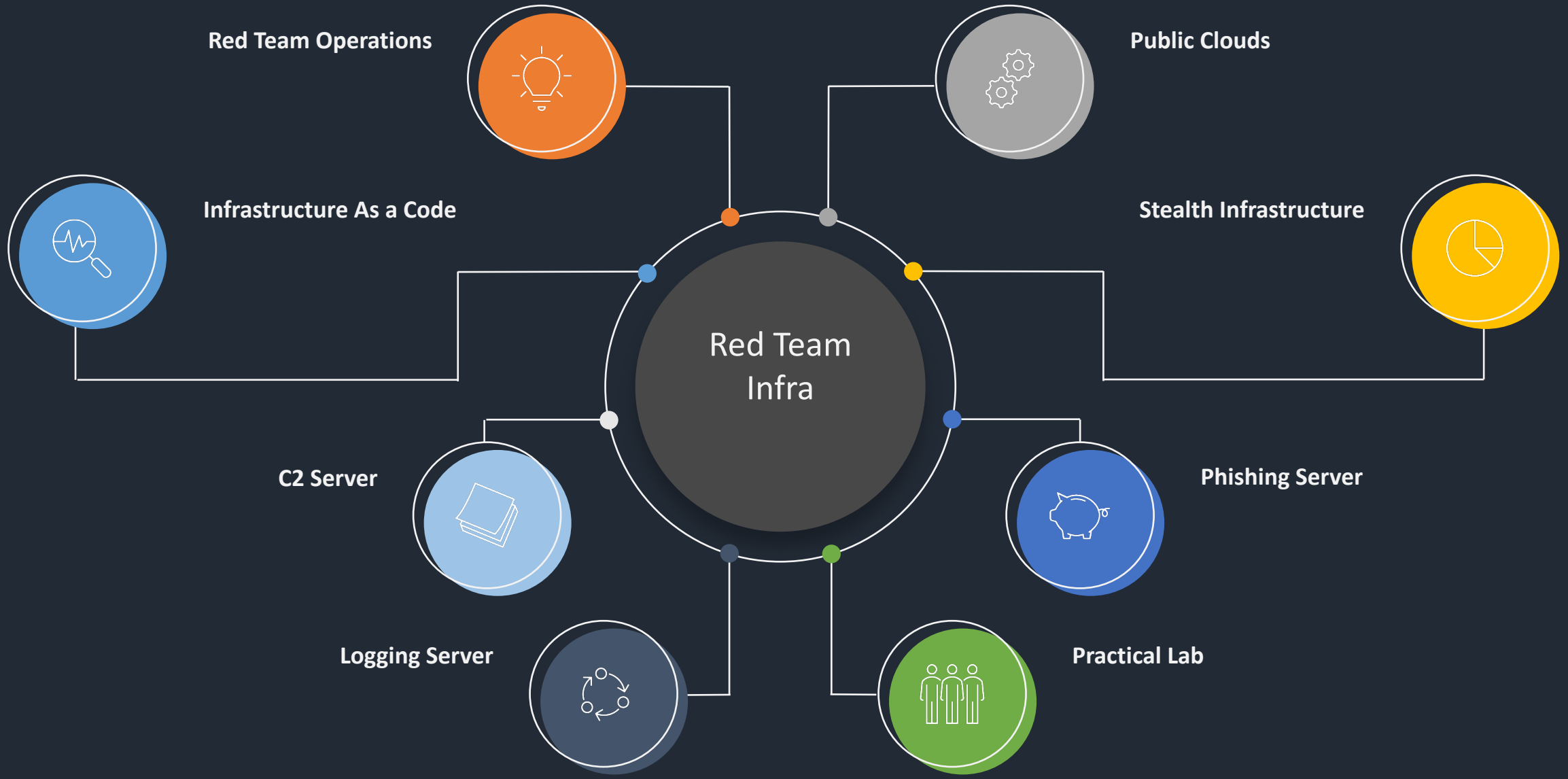
Agenda


Red Team Operations
Public Clouds
Infrastructure As a Code
Stealth Infrastructure
Red Team Infra
C2 Server
Phishing Server
Logging Server
Practical Lab

About Me (Arun Nair)

Full time memer, part time hacker

Go by the name Dazzy Ddos (@dazzyddos)

A hardcore anime fan

Explorer at heart, thrives on discovering new places - ChatGPT ;)

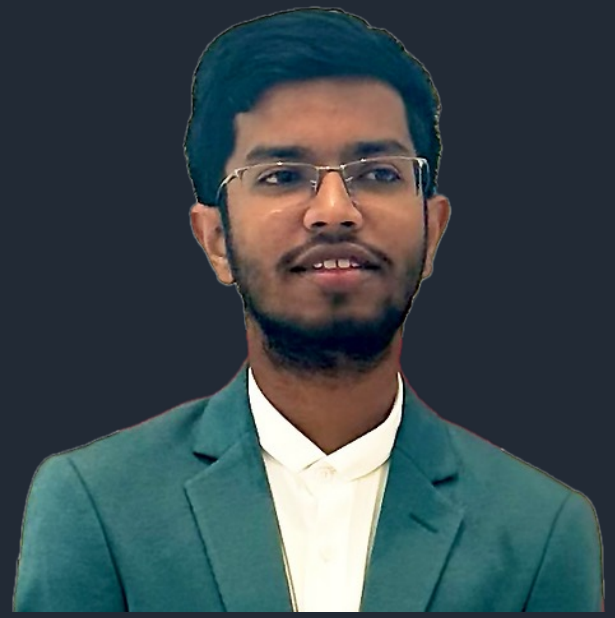
About Aravind

Red Teamer @Resillion

Wanna be malware dev

Avid motorcyclist, always on the trail
for the next long ride – lol ChatGPT again

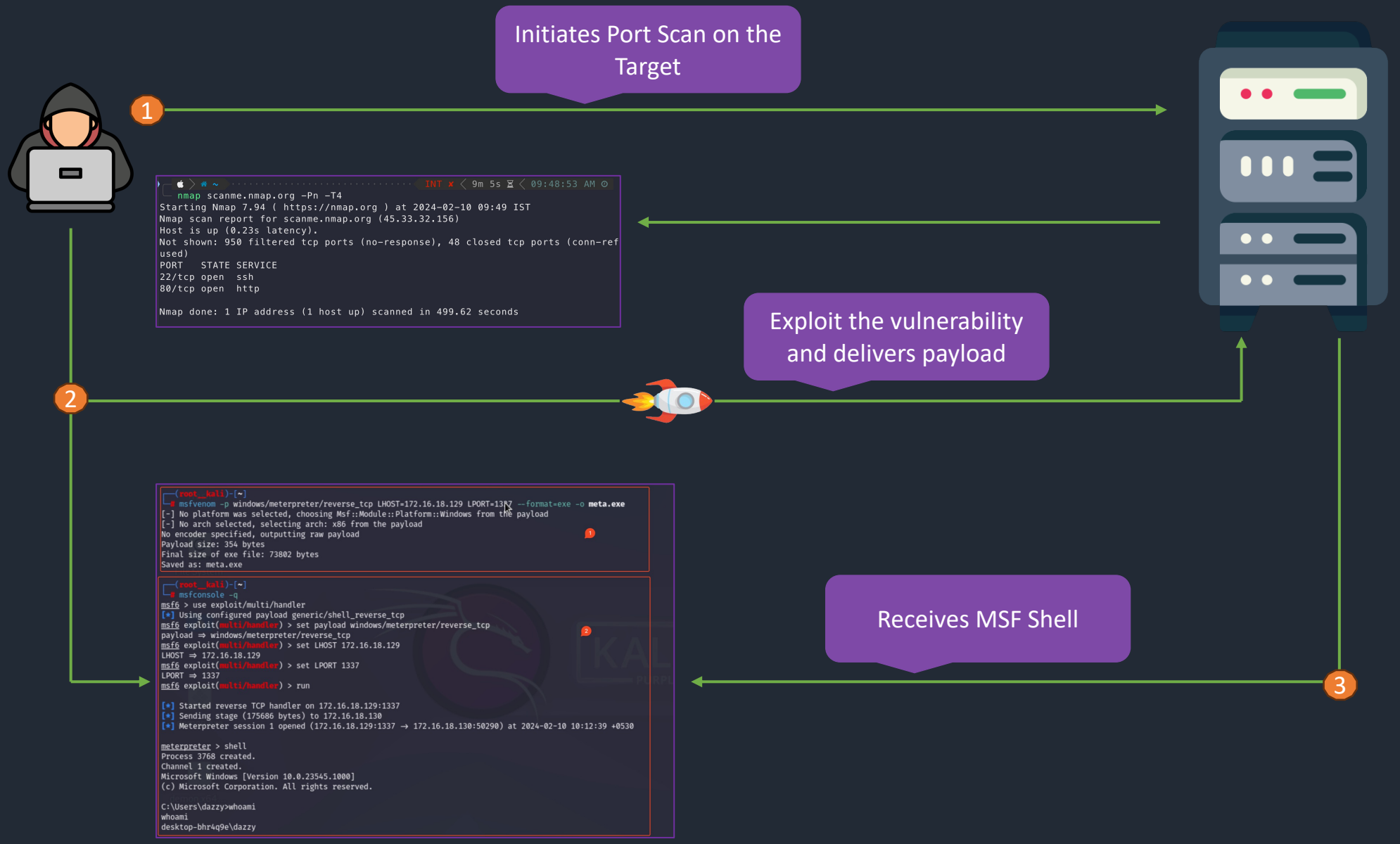
About Soumyadeep



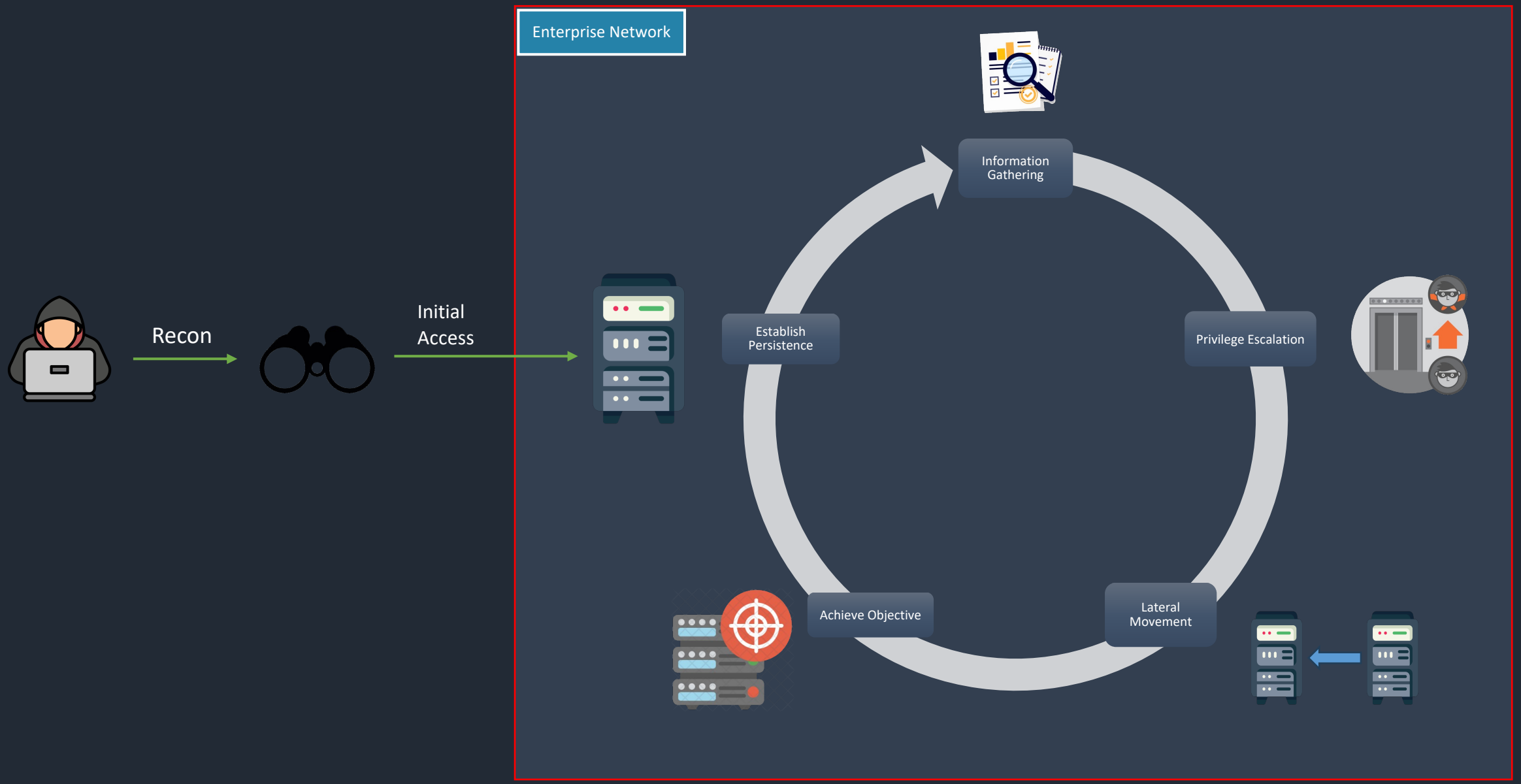
Cloud Threat Detection Engineer @CRED

Spoken at multiple conferences (ringzer0, c0c0n etc)

Ya'll Still Play CTF?



Red Team Phases



Fitting Infrastructure Components to Phases

Reconnaissance Phase

- Attack Box (Proxy Server) is utilized for gathering intelligence and enumerating the target environment. It acts as a proxy to conceal the operator's true IP address, allowing for IP rotation in case of blocking

Initial Access Phase

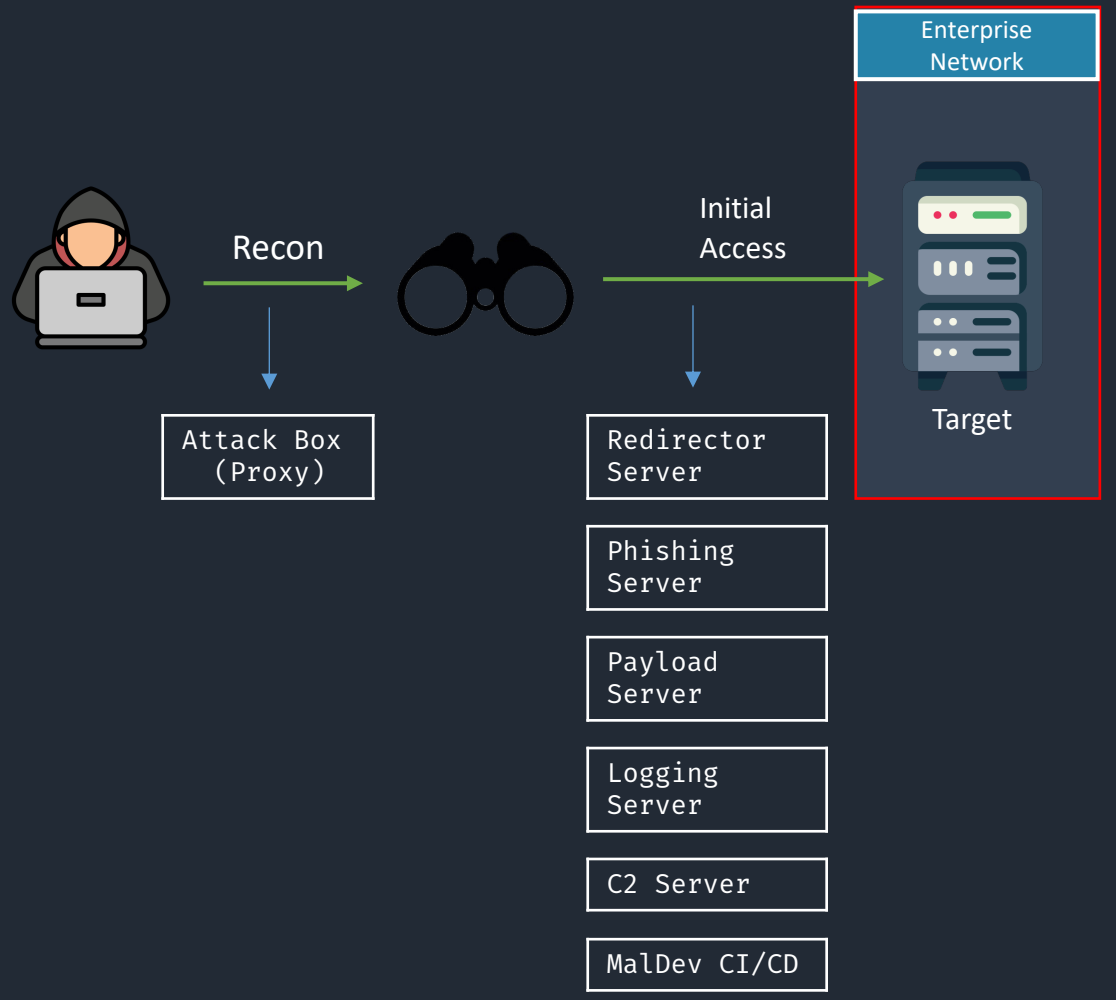
- Redirector Server masks traffic to the C2 server
- Phishing server hosts phishing pages and collects credentials
- Payload server distributes malware payloads after initial contact is made
- Logging server records operational data and interactions with target for analysis and operational awareness
- C2 Server manages compromised systems, sends commands and receives callbacks from the target network

Post-Compromise Phase

- The C2 server continues to play a central role, now for maintaining control over the compromised systems, executing follow-up commands and exfiltrating data
- The Logging Server aids in maintaining a detailed record of the actions taken and data exfiltrated during this phase

Maintenance and Scaling

- Throughout the operation, the Redirector Server's role evolves to manage and distribute incoming and outgoing traffic effectively, adapting to any changes in the target's defense posture
- Attack Box may be used for additional reconnaissance or to manage other components of the infrastructure based on the gathered intelligence



Red Team Infrastructure Design Essentials

SCALABLE

The infrastructure should be ready to expand or change as needed without missing a beat

SECURITY

All data must be encrypted as it moves in and out and there should be a system to monitor and log activities

STREAMLINED

Infrastructure should leverage automation tools for efficient setup, tear-down and management

Traditional v/s Modern Infrastructure

	Traditional	Modern
Bastion Host	✗	✓
Smart Redirectors	✗	✓
Logging Server	✗	✓
Cloud Services Integration	✗	✓
Opsec Practices	✗	✓
Automated Tools and Scripts	✗	✓

- ✓ Modern infrastructure uses bastion hosts, also known as jump boxes, to provide a secure and controlled entry point to access internal servers
- ✓ Modern red team infrastructures integrate smart redirectors to intelligently handle incoming traffic
- ✓ Modern infrastructures incorporate logging servers to aggregate logs from various sources, enabling better monitoring of operations and aiding in the detection of blue team activities.
- ✓ Modern infrastructures are cloud-integrated, taking advantage of the scalability, global distribution, and on-demand resources that cloud providers offer
- ✓ Modern red team infrastructures heavily use automation, with tools and scripts that streamline the deployment, management, and execution of operations, increasing efficiency and reducing the chance of mistakes.

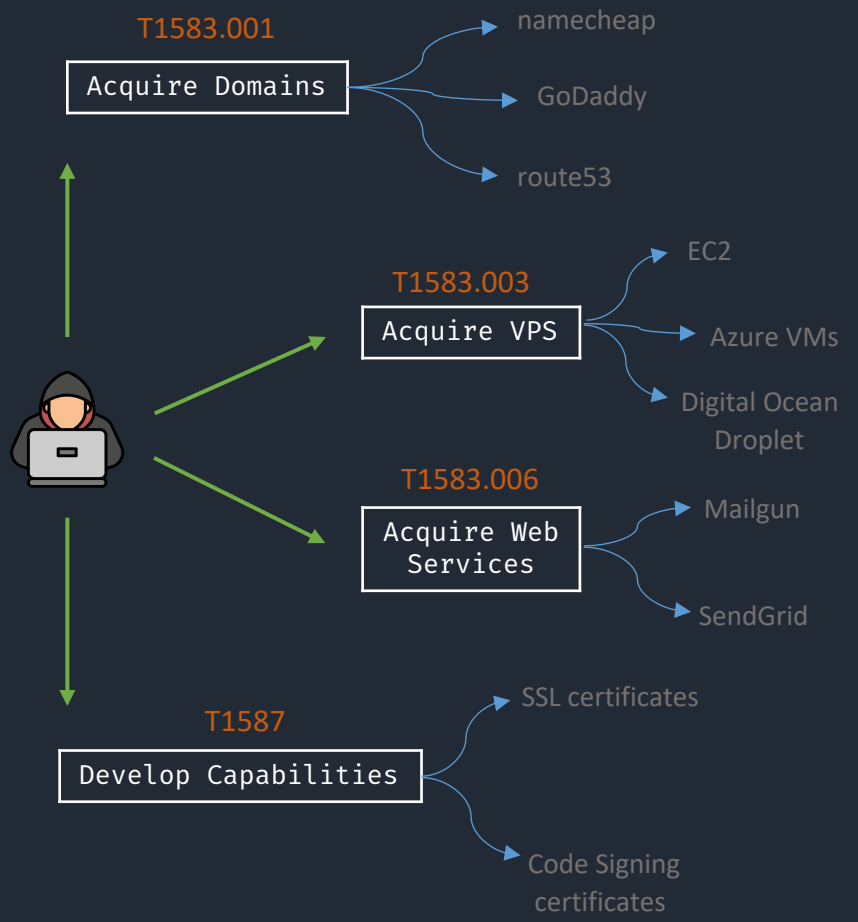
Prerequisites for Red Team Infrastructure

Domain Acquisition
- Importance of domain names for legitimate-looking traffic
- Selection of domain names aligned with the target environment
- Use of non-attributable methods for registration to maintain opsec

Infrastructure Acquisition
- Choosing right mix of cloud and physical servers
- Diversification across providers like Amazon EC2, Azure VMs etc
- Leveraging CDNs for distribution and resilience

Secure Communication Setup
- Procurement of digital certificates for encrypted traffic (HTTPs)
- Use of code signing certificates with Payload for evasion

Operational Security
- Ensuring purchases and registrations are untraceable
- Implementation of access control and MFA on critical endpoints
- Utilizing relays and redirectors to mask true infrastructure



https://attack.mitre.org/tactics/TA0042/
https://scythe.io/library/attack-infrastructure-red-teams-vs-malicious-actors

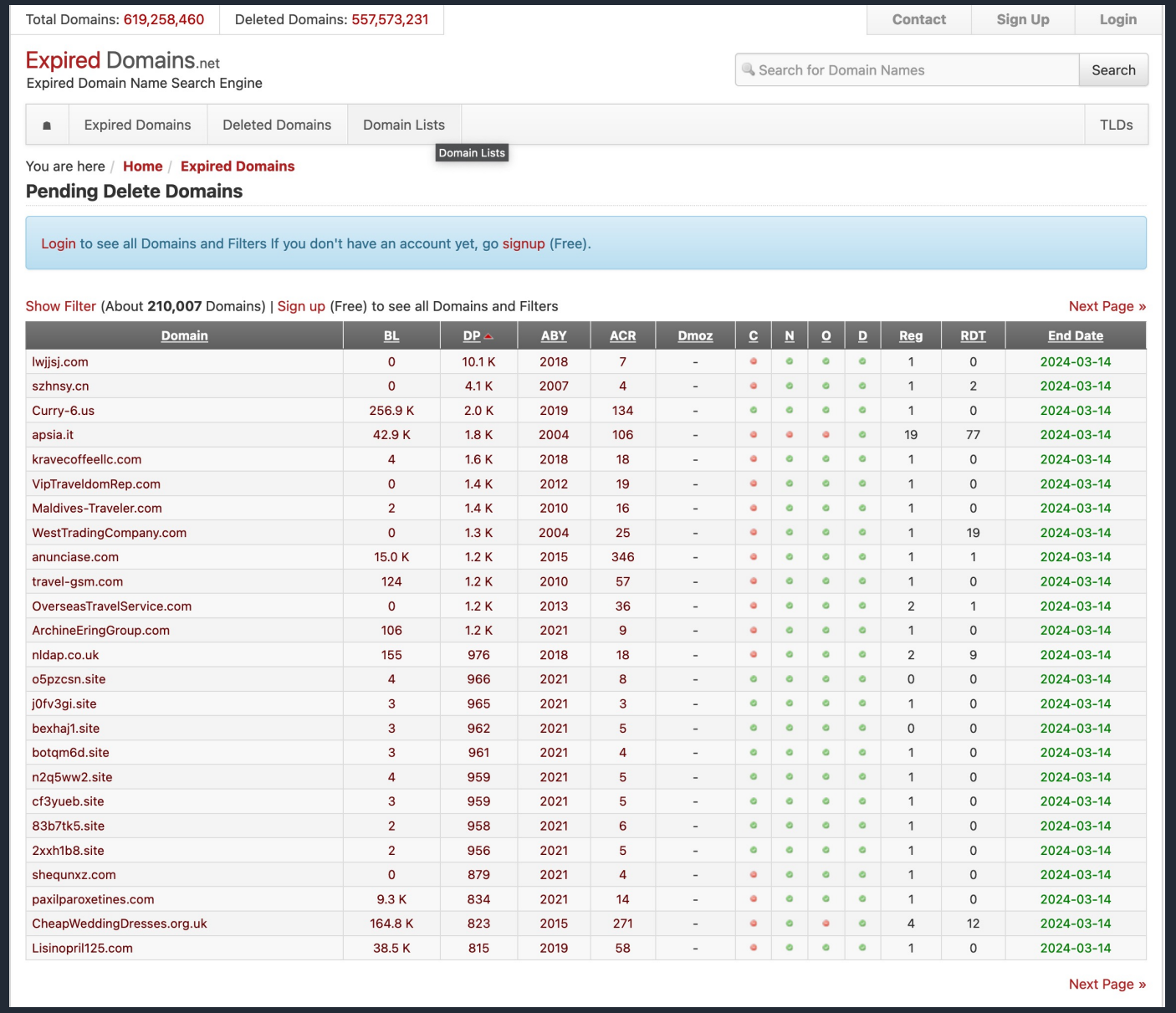
Expired Domains

Comprehensive database of expired, dropping, and available domains

Offers a variety of filters to help for domains by specific criteria such as domain age, PageRank, backlinks, traffic, and more

Using domains with existing history and backlinks can increase the credibility and chances of phishing mails landing in target's inbox

Can quickly acquire a domain that matches the target organization's profile and can accelerate the setup phase of red teaming exercises



Domain	BL	DP	ABY	ACR	Dmoz	C	N	O	D	Reg	RDT	End Date
lwjjsj.com	0	10.1 K	2018	7	-					1	0	2024-03-14
szhnsy.cn	0	4.1 K	2007	4	-					1	2	2024-03-14
Curry-6.us	256.9 K	2.0 K	2019	134	-					1	0	2024-03-14
apsia.it	42.9 K	1.8 K	2004	106	-					19	77	2024-03-14
kravecoffeellc.com	4	1.6 K	2018	18	-					1	0	2024-03-14
VipTraveldomRep.com	0	1.4 K	2012	19	-					1	0	2024-03-14
Maldives-Traveler.com	2	1.4 K	2010	16	-					1	0	2024-03-14
WestTradingCompany.com	0	1.3 K	2004	25	-					1	19	2024-03-14
anunciase.com	15.0 K	1.2 K	2015	346	-					1	1	2024-03-14
travel-gsm.com	124	1.2 K	2010	57	-					1	0	2024-03-14
OverseasTravelService.com	0	1.2 K	2013	36	-					2	1	2024-03-14
ArchineEringGroup.com	106	1.2 K	2021	9	-					1	0	2024-03-14
nldap.co.uk	155	976	2018	18	-					2	9	2024-03-14
o5pzcsn.site	4	966	2021	8	-					0	0	2024-03-14
j0fv3gi.site	3	965	2021	3	-					1	0	2024-03-14
bexhaj1.site	3	962	2021	5	-					0	0	2024-03-14
botqm6d.site	3	961	2021	4	-					1	0	2024-03-14
n2q5ww2.site	4	959	2021	5	-					1	0	2024-03-14
cf3yueb.site	3	959	2021	5	-					1	0	2024-03-14
83b7tk5.site	2	958	2021	6	-					1	0	2024-03-14
2xxh1b8.site	2	956	2021	5	-					1	0	2024-03-14
shequnxz.com	0	879	2021	4	-					1	0	2024-03-14
paxilparoxetines.com	9.3 K	834	2021	14	-					1	0	2024-03-14
CheapWeddingDresses.org.uk	164.8 K	823	2015	271	-					4	12	2024-03-14
Lisinopril125.com	38.5 K	815	2019	58	-					1	0	2024-03-14

https://expireddomains.net

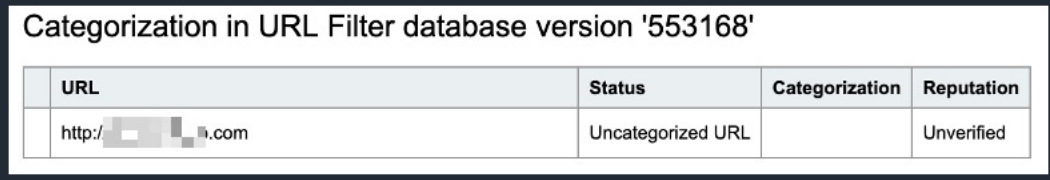
Domain Categorization

Just like the new kids have a hard time fitting in, new domains often get blocked because they're not recognized or trusted yet

Some organization proxies only let in the well-known or 'elite' categories' domains, like finance or government sites.

Some clever strategies for Domain Categorization would be:

- **Revive an Old Domain** – Snap up domains that just left the game. An expired domain with a history could be re-used to slip past the proxies unnoticed
- **Categorize Your Domain** – Clone an existing legit-looking website and submit it for categorization
- **Camouflage with a categorized site (cloud services)** – Hide in plain sight by piggybacking on cloud services' domain (Azure CDN, AWS Lambda)



Categorization in URL Filter database version '553168'

	URL	Status	Categorization	Reputation
	http:/ .com	Uncategorized URL		Unverified

Some sites for checking/submitting domain categorization

Bluecoat/Symantec - https://sitereview.bluecoat.com/#/
McAfee - https://www.trustedsource.org
Palo Alto Wildfire - https://urlfiltering.paloaltonetworks.com
Websense - https://csi.forcepoint.com & https://www.websense.com/content/SiteLookup.aspx (needs registration)
FortiGuard - https://www.fortiguard.com/webfilter
IBM X-force - https://exchange.xforce.ibmcloud.com
Cyren - https://www.cyren.com/security-center/url-category-check-gate
Checkpoint - https://www.checkpoint.com/urlcat/main.htm (needs registration)

Before vs After Categorization

URL: [redacted]

Categories: Parked

Risk Level: Low-Risk

Category: Parked

Description: URLs which host limited content or click-through ads which may generate revenue for the host entity but generally do not contain content that is useful to the end user

Example Sites: www.parked.com

Risk Level: Low-Risk

Description: Any site that is not High Risk or Medium Risk. This includes sites that were previously confirmed as malicious but have displayed benign activity for at least 90 days

Example Sites: www.google.com, www.schwab.com, www.amazon.com

Request Change

no-reply-url-feedback@paloaltonetworks.com

to me

Thanks again for your URL re-categorization request. As a result of our re-evaluation, we have made the following changes:

URL: [redacted]
Previous category: parked
You suggested: business-and-economy
Accepted category: business-and-economy

URL: [redacted]

Categories: Business-and-Economy

Risk Level: Low-Risk

Category: Business-and-Economy

Description: Marketing, management, economics, and sites related to entrepreneurship or running a business. Includes advertising/marketing firms as well as shipping site such as fedex.com and ups.com. Corporate websites might be categorized with their technology instead of this category

Example Sites: www.bothsidesofthetable.com/, www.ogilvy.com, www.geisheker.com/, www.imageworksstudio.com/, www.linearcreative.com/

Risk Level: Low-Risk

Description: Any site that is not High Risk or Medium Risk. This includes sites that were previously confirmed as malicious but have displayed benign activity for at least 90 days

Example Sites: www.google.com, www.schwab.com, www.amazon.com

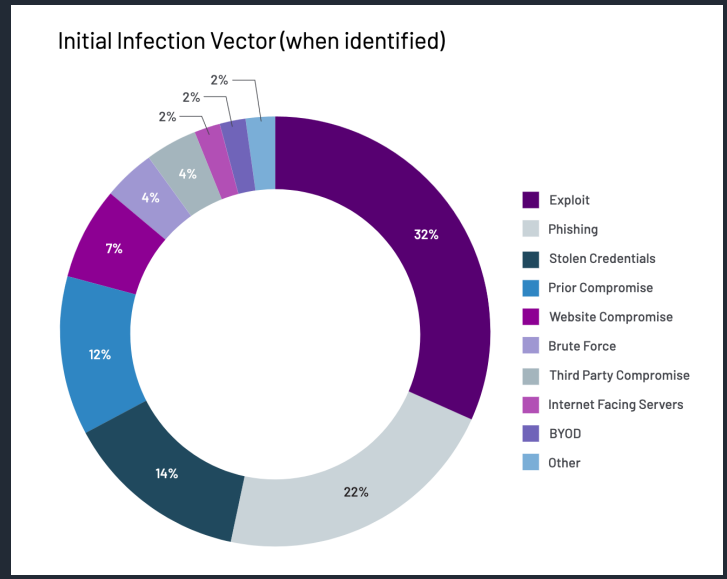
Request Change

Persistent Threat of Phishing

Phishing continues to be the most common initial access vector for cybersecurity incidents.

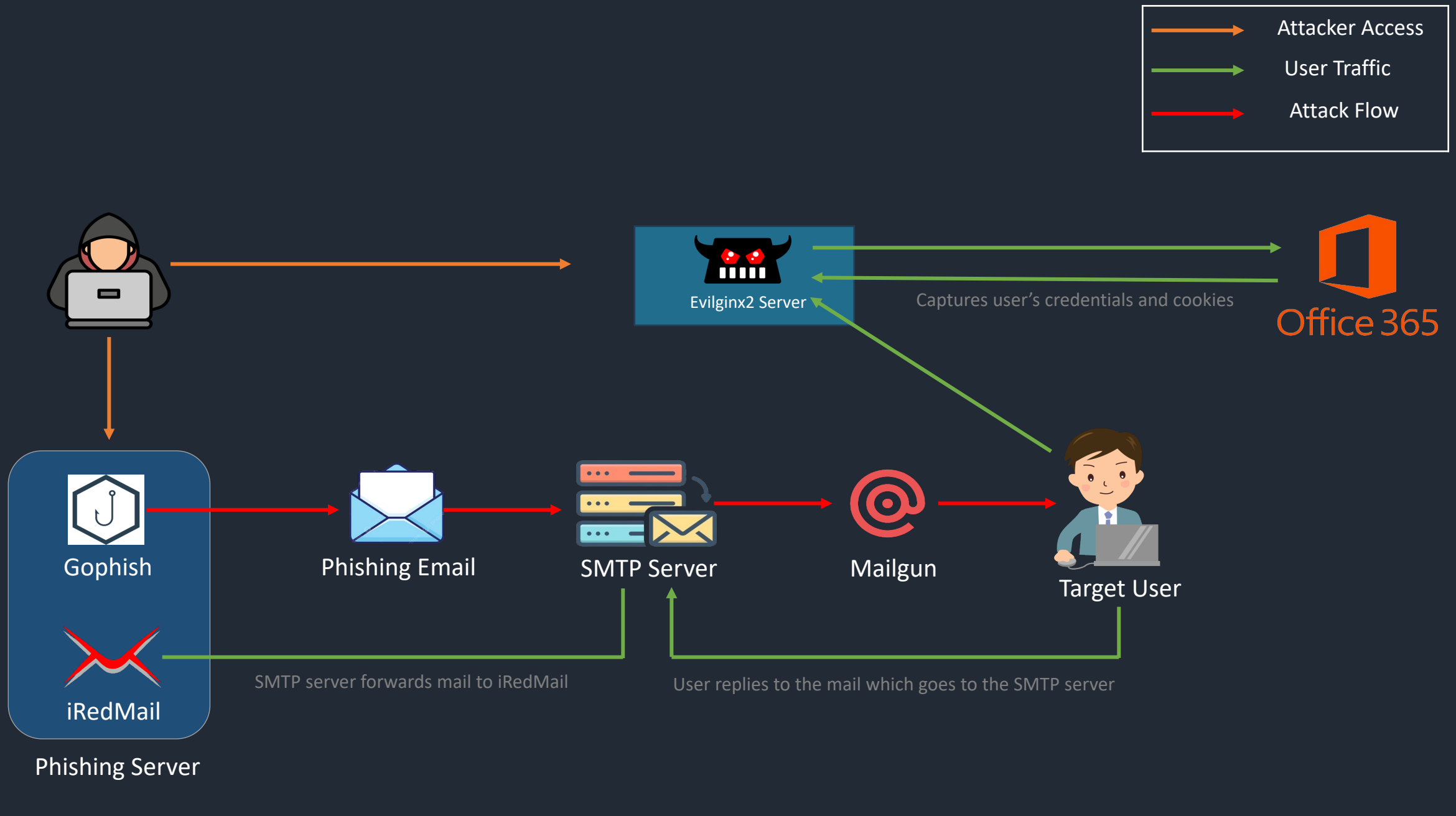
Phishing has been a go-to method for getting into systems for a long time. Hackers keep coming up with new tricks to make sure phishing stays effective.

- https://www.bleepingcomputer.com/news/security/microsoft-teams-phishing-pushes-darkgate-malware-via-group-chats/
- https://jeffreyappel.nl/aitm-mfa-phishing-attacks-in-combination-with-new-microsoft-protections-2023-edt/

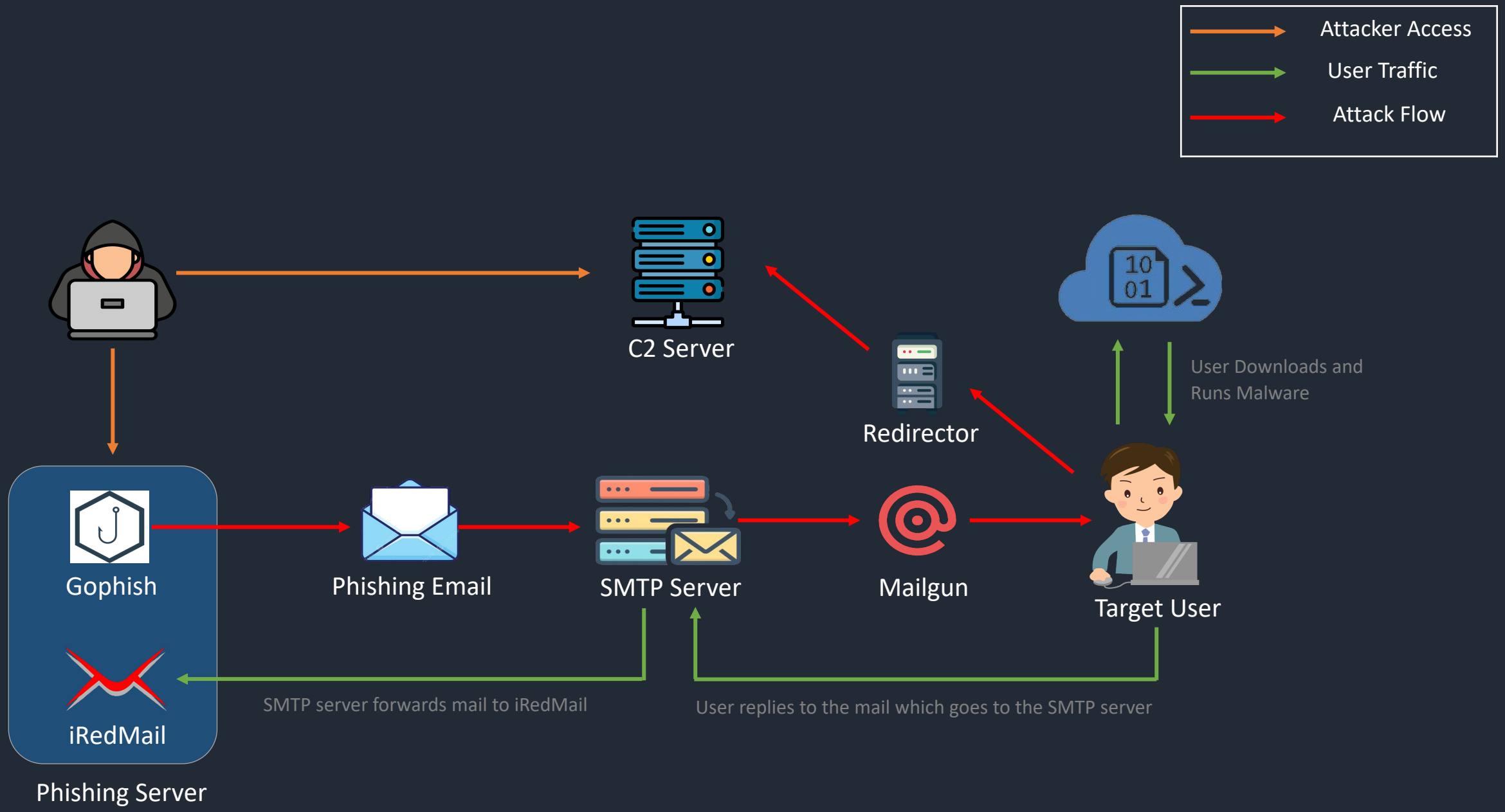


mandiant.com/m-trends

Phishing Setup for Credentials

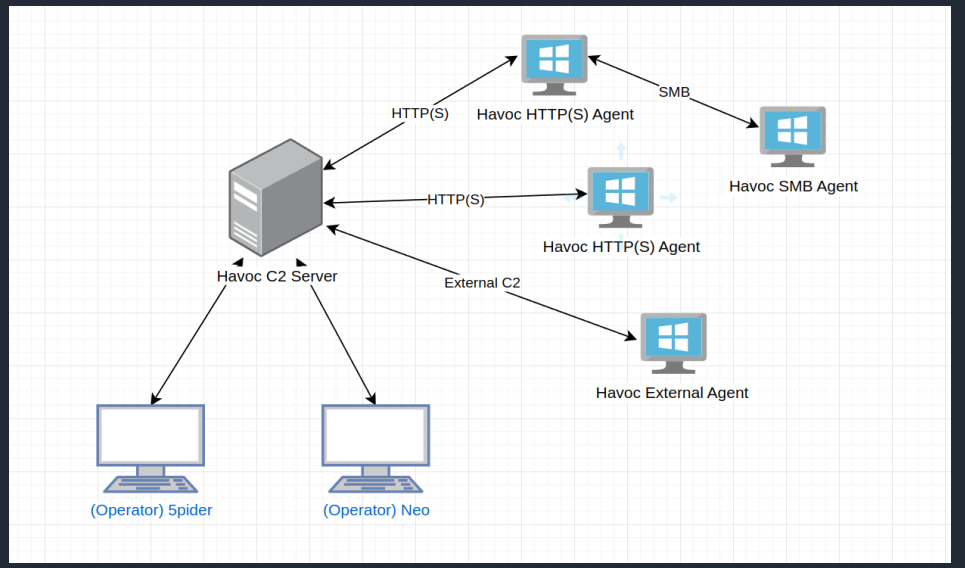


Phishing Setup for Malware



Havoc C2 Overview

- Open Source C2 Framework by @C5pider (Paul Ungur) 
- Well documented at https://havocframework.com/docs/welcome
- Havoc operates through a two-part architecture:
  - **Teamservers:** These are the nerve centers of Havoc, managing connections from operators and routing instructions to agents. They are responsible for parsing callbacks, managing listeners, and handling files or screenshots received from agents. Teamservers are typically hosted on public Virtual Private Servers (VPS) to ensure accessibility for authenticated operators.
  - **Clients:** These serve as the interface for the Teamservers, offering a platform for operators to issue commands to agents and review the resulting data or outputs.



Source: https://havocframework.com/docs/welcome

GitHub Workflows

Automate CI/CD Pipeline in Github

Triggered by events like push, pull requests, or scheduled times

Defined by YAML files in the .github/workflows directory

```yaml
name: Update on Pull Request

on:
  pull_request:
    branches:
      - main
jobs:
  update-third-party-content:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout our repository
      uses: actions/checkout@v3

    - name: Clone third-party repository
      run: |
        git clone https://github.com/external/repo.git myrepo

    - name: Make changes in the third-party repository
      run: |
        echo “Updated” >> myrepo/important-file.txt

    - name: Commit and push to our repository
      run: |
        git config user.name "Your GitHub Username"
        git config user.email "your.email@example.com"
        git add ./external-updates/
        git commit -m "Updated with latest changes"
        git push origin HEAD:main
```

Risks of Hosting Red Team Infrastructure On the Internet

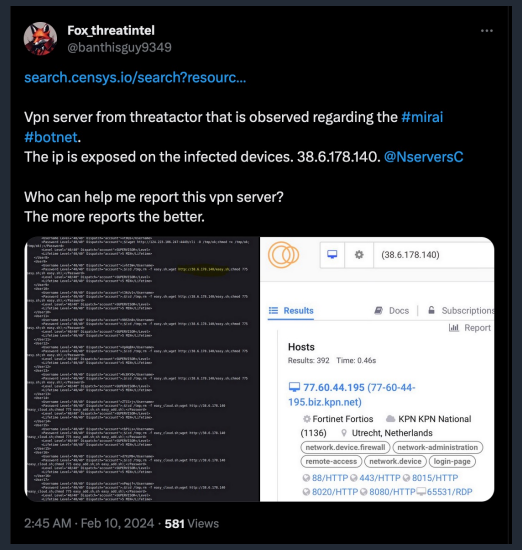
Directly hosted servers are easily detectable by automated scanning tools and defensive monitoring systems, making them prime targets for blacklisting and vulnerability scanning

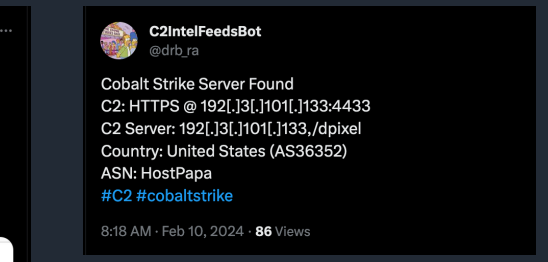
Blue Teamers can easily identify the infrastructure, leading to early detection and potential engagement closure

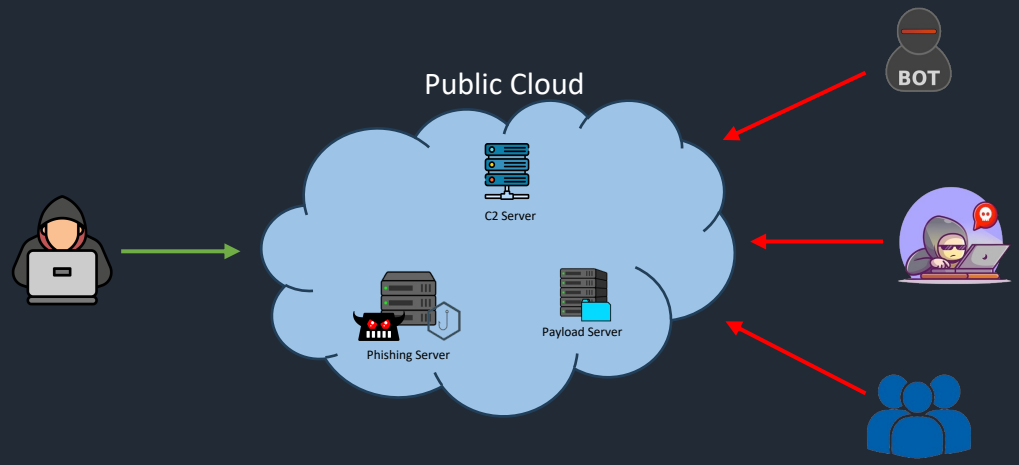
Traffic patterns to and from these servers can be analysed more straightforwardly, revealing malicious activities and the nature of the operation

Without redirectors, the IP addresses of our infrastructure can be quickly traced back to us or our organization, compromising anonymity and operational security.

Directly exposed infrastructure presents a single point of failure, risking the entire operation if any component is compromised or taken down







Hunting Malicious Infrastructure 101

Pattern Recognition

- Analysts identify common configuration patterns across servers, unique identifiers in TLS certificates or service banners are catalogued. For instance, a certificate with the name 'AsyncRAT Server' or a service banner containing 'X-Havoc' can be indicative of malicious use.
- Utilizing threat intelligence databases and public IOC repositories to cross-reference suspicious attributes

Network Behaviour

- Monitoring network traffic for unusual patterns or beaconing to known bad domains or IP addresses
- Using baselines to detect deviations in network behaviour that could indicate C2 communications or data exfiltration

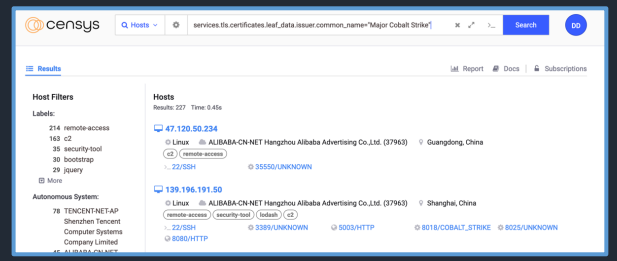
Infrastructure Footprints

- Repeated use of specific hosting services or Autonomous System Numbers (ASNs) associated with previous malicious activity can reveal new nodes of infrastructure
- Open directories that host suspicious files are flagged. Common malicious file names or tools like 'nc.exe' or 'procdump.exe' are searched across the web

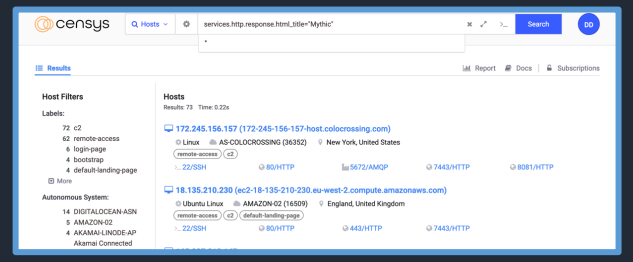
Some of the most common indicators that threat actors will re-use are:

- **Certificate Information** – Fields inside of TLS and SSL certificates. Hardcoded values are often re-used.
- **Server Headers** – Actors deploying custom software may forget to change default headers that contain indicators.
- **Data in HTTP Responses** – Custom software containing unique values in HTTP responses
- **Location, ASN and Hosting Providers** – Actors re-using hosting providers for infrastructure. Similar servers may be hosted at the same ASN.
- **JA3 Hashes** – Actors deploying uncommon software configurations can be fingerprinted by JA3 signatures.
- **Port Configurations** – Actors will often leave the same ports open across infrastructure.
- **Regular Expressions** – Actors may deploy unique values with highly similar structure that can be captured with Regular Expressions.

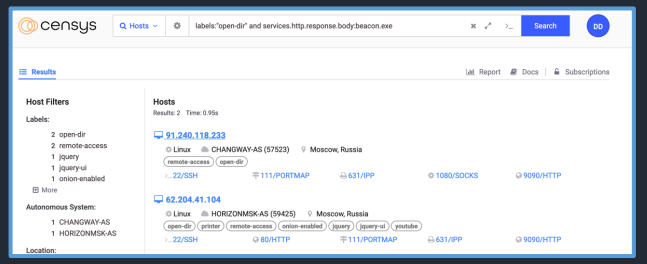
Hunting Cobalt Strike with TLS Certificate

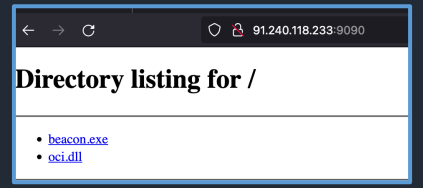


Hunting Mythic with HTTP Response Title



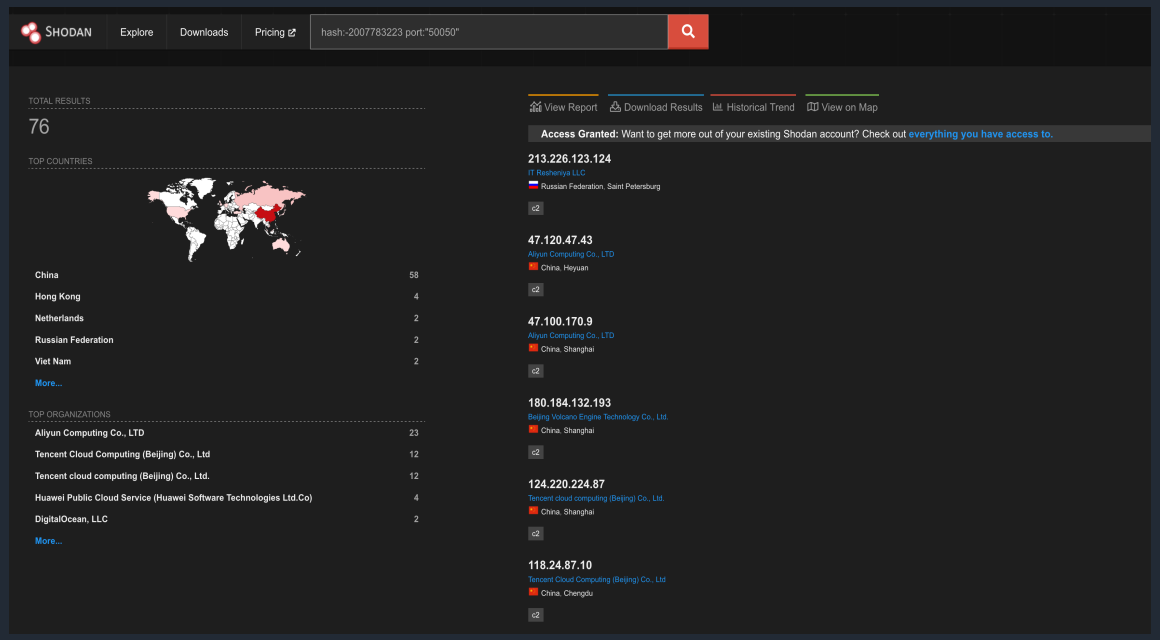
Hunting Cobalt Strike with Open Directories

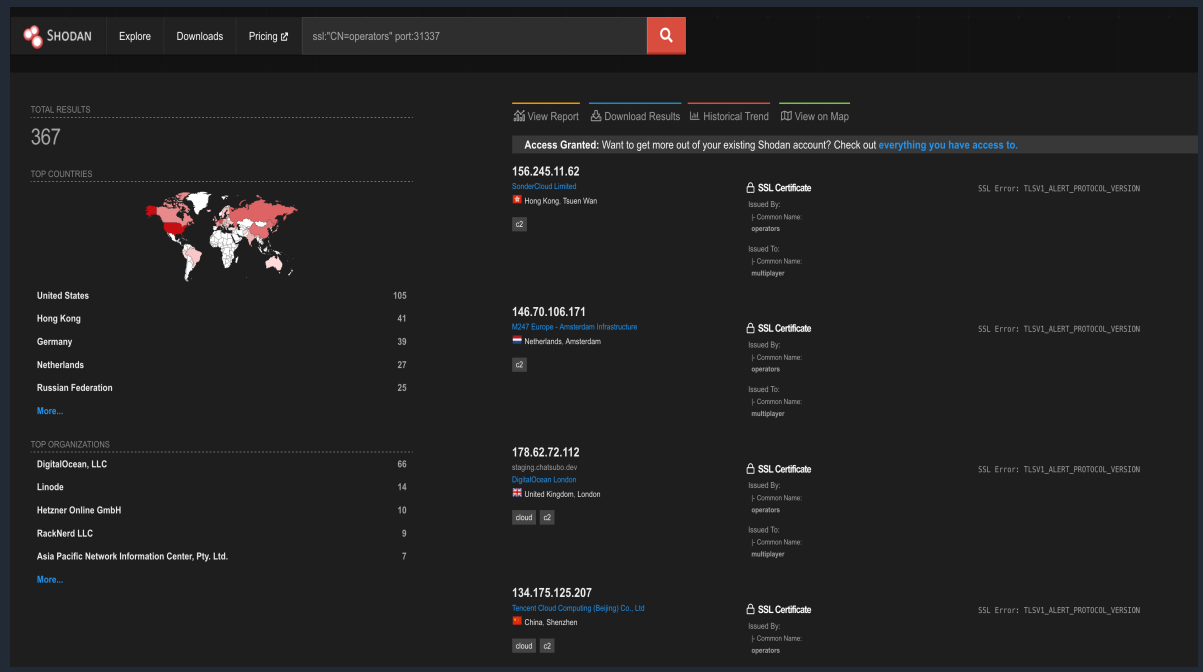


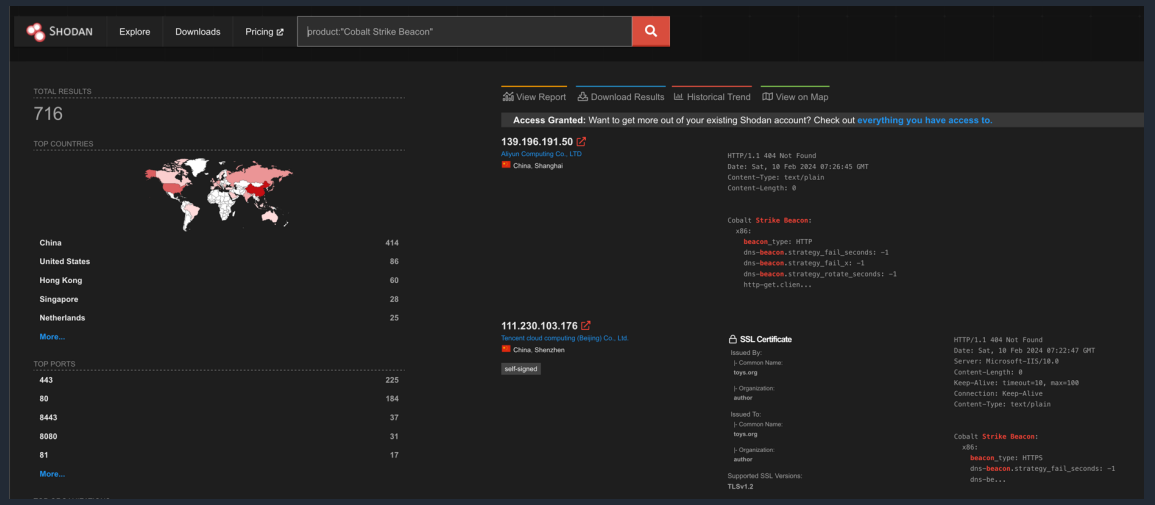


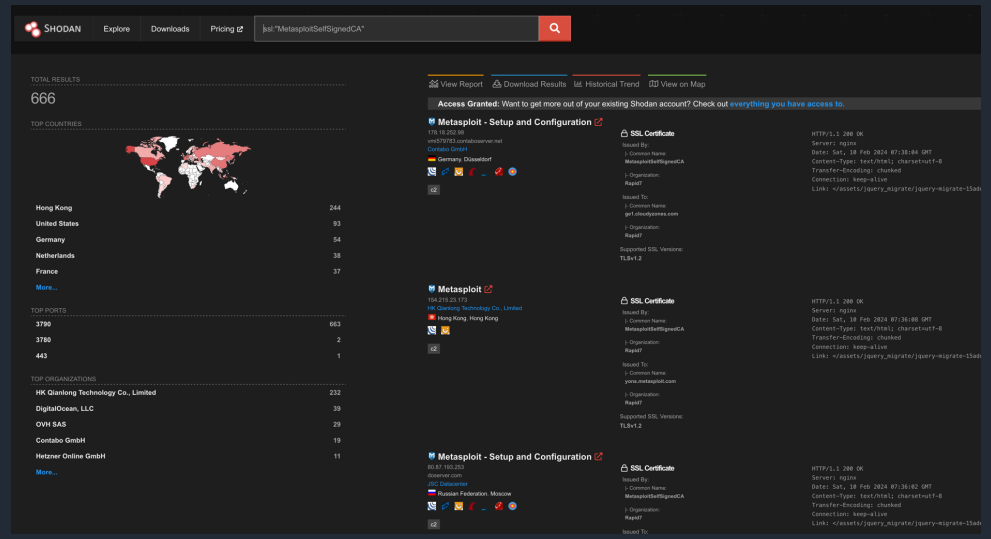
Source: https://censys.com/a-beginners-guide-to-tracking-malware-infrastructure/

Wanna Find Some More C2 Servers?



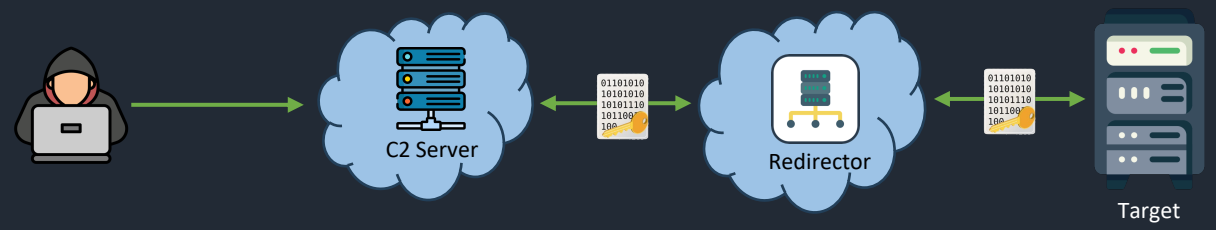




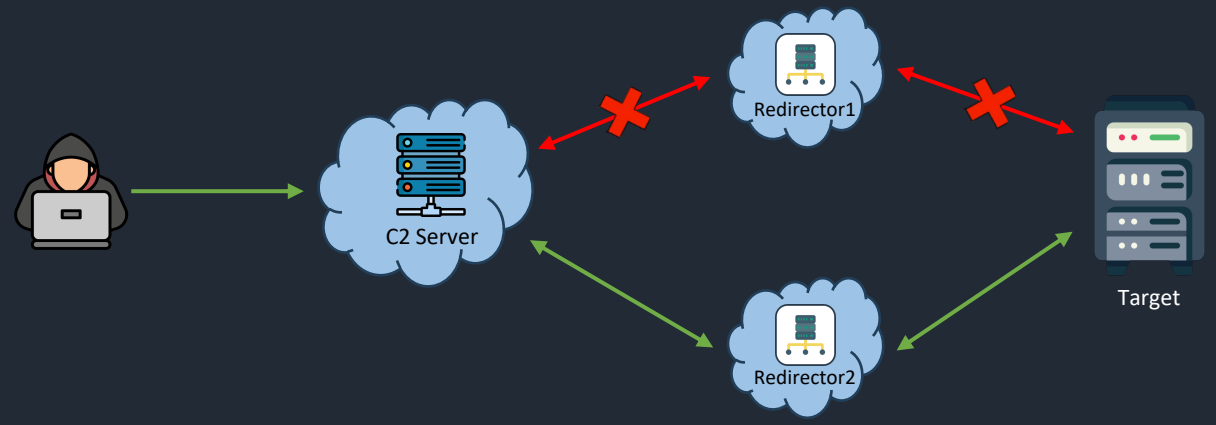


Redirectors to the Rescue

Redirectors obfuscate the source and nature of the malicious traffic, such as SSL/TLS encryption, domain fronting, or mimicking legitimate web traffic, redirectors can blend in with normal internet traffic, evading signature-based detection systems.

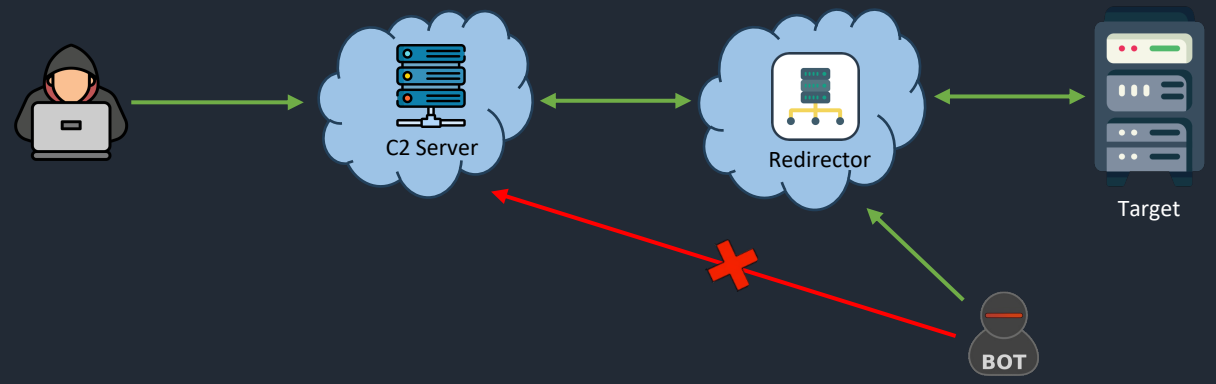


By using multiple redirectors, red teams can create a more resilient infrastructure that can quickly adapt if one node is compromised or identified as redirectors are considered burnable component



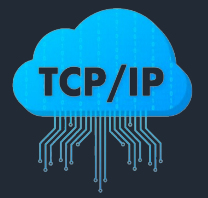
Redirector IPs and domains can be rotated to evade blacklists and, making it difficult for automated scans to keep up.

Redirectors serve as an intermediary layer, meaning that bots and scanners encounter the redirector instead of the actual C2 server. If a redirector is detected, its purpose can remain ambiguous, as it doesn't hold any tools or payloads directly associated with Red Team activities.



Redirector Goals

Purpose	Designed to handle specific types of traffic (HTTPS, SMTP, DNS) and only allow desired traffic to reach the private infrastructure, like C2 server or payload server
Deception	They can be configured to mislead investigators by controlling the outward appearance of the attack infrastructure
Cloud-Based	Typically hosted on cloud platforms for scalability. Easy to decommission and spin up
Lightweight and Minimalistic	Require minimal setup; often only a basic web server is necessary
Easily Automated	Can be quickly deployed or destroyed via automation tools, ideal of dynamic operation operations



Types of Redirectors

Basic Redirectors

Also known as blind reverse proxies, such as ”socat” or “iptables”, which offer simple forwarding with minimal filtering and setup effort

socat example to redirect all traffic to 10.1.1.1:80

```
socat TCP4-LISTEN:80, fork TCP4:10.1.1.1:80
```

Achieving the same above objective using iptables (more complex)

```
iptables –I INPUT –p tcp ––dport 80 –j ACCEPT
iptables -t nat -A PREROUTING -p tcp --dport 80 -j
DNAT --to-destination 10.1.1.1:80
iptables -t nat -A POSTROUTING -j MASQUERADE
iptables -I FORWARD -j ACCEPT
iptables -P FORWARD ACCEPT
sysctl net.ipv4.ip_forward=1
```

These redirectors pass most(all) traffic to the private infrastructure which increases the likelihood of identification and categorization of the redirector as malicious

Advanced Redirectors

Reverse proxies with complex rules for granular traffic control, such as Apache configured to only allow specific URI or User Agent strings through a back C2 server or allow only IP addresses from specific countries.

Below is a sample Apache configuration that checks whether the URI has ”/rofl/”, if not it will redirect the request to google.com

```
<VirtualHost *:80>

DocumentRoot /var/www/html/

RewriteEngine On
RewriteCond %{REQUEST_URI} ^/rofl/(\S+)$
RewriteRule ^.*$ http://10.1.1.1:80%{REQUEST_URI}
[P]
ProxyPassReverse ^ http://10.1.1.1:80
# If the above rules don't match, send to Google.
RewriteRule ^.*$ http://www.google.com/? [L,R=302]

</VirtualHost>
```

CDNs as a Redirector

CDNs are typically used to cache content, improving access speed

Advantages of CDN as a Redirector

Global Presence – CDNs have a widespread network of servers, making it harder to pinpoint the origin of traffic

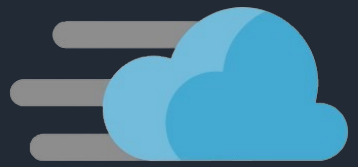
Scalability – Easily and automatically scales to handle large volumes of traffic

Custom Domains – Mask the origin of content by using custom domains

Reputation Benefits – Utilizing CDN allows to piggyback on the trust and reputation of the Cloud provider's domain space, often whitelisted by organizations

SSL\TLS Encryption – CDN supports encryption out of the box

AWS CloudFront



Azure CDN

CloudFlare

Apache as an HTTP Redirector

Apache is a widely used open-source web server that can serve web content and can also act as a reverse proxy

Apache can be configured to route incoming traffic to different backend servers based on rules and conditions, such as URL paths or source IP addresses.

We will be utilizing Apache's "mod_rewrite" and related modules to forward requests to backend servers

Apache provides detailed access and error logs, which can be crucial for monitoring redirection and identifying potential blue team activities on our infrastructure

Apache conf files exist in **/etc/apache2/sites-available**

httpredir.conf

```
<VirtualHost *:80>

DocumentRoot /var/www/html/

Include /etc/apache2/modmaxmind.conf

RewriteEngine On
RewriteCond %{REQUEST_URI} ^/rofl/(\S+)$
RewriteRule ^.*$ http://10.1.1.1:80%{REQUEST_URI} [P]
ProxyPassReverse ^ http://10.1.1.1:80
# If the above rules don't match, send to Google.
RewriteRule ^.*$ http://www.google.com/? [L,R=302]

</VirtualHost>
```

modmaxmind.conf

```
MaxMindDBEnable On
MaxMindDBFile COUNTRY_DB
/usr/local/share/maxminddb/GeoLite2-Country.mmdb
MaxMindDBEnv MM_COUNTRY_CODE
COUNTRY_DB/country/iso_code

SetEnvIf MM_COUNTRY_CODE ^(COUNTRY_LIST) AllowCountry
<Location "/">
Deny from all
Allow from env=AllowCountry
</Location>
```

```
--- # Setup HTTP Redirector
- hosts: all
  become: yes
  tasks:
  - name: Install Apache2
    apt:
      name: apache2
      state: present

  - name: Enable Apache2 mod_rewrite
    apache2_module:
      name: rewrite
      state: present

  - name: copy conf file
    copy:
      src: /tmp/httpredir.conf
      dest: /etc/apache2/sites-
available/httpredir.conf

  - name: Enabling our configuration
    shell:
    cmd: a2ensite
/etc/apache2/httpredir.conf

  - name: Reload Apache
    service:
      name: apache2
      state: reloaded
```

URL Manipulation with mod_rewrite

A built-in module for Apache that enables URL rewriting

It provides rule-based redirections which direct traffic based on a variety of conditions like source IP, requested URI or browser User-Agent strings.

Use Cases for Redirector

Traffic Filtering - Only allow requests with specific characteristics (e.g., headers, user agents) to reach sensitive endpoints.

Deception - Mislead automated scanners and attackers by serving different content based on their request patterns.

Operational Security - Hide the true nature of your red team infrastructure by dynamically changing request paths and destinations.

Conditional Redirection Based on User-Agent

```
RewriteEngine On
RewriteCond %{HTTP_USER_AGENT} googlebot [NC]
RewriteRule ^(.*)$ /crawler-page.html [L]
```

Blocking Traffic by IP

```
RewriteEngine On
RewriteCond %{REMOTE_ADDR} ^123\.456\.789\.000$
RewriteRule .* - [F]
```

Conditional Redirection Based on Referer Header

```
RewriteEngine On
RewriteCond %{HTTP_REFERER} somespecificreferrer\.com [NC]
RewriteRule .* /special-offer.html [L]
```

mod_rewrite uses the following directives:

- **RewriteEngine –** Activates or deactivates the runtime rewriting engine. It's typically set to "On" to enable rewriting rules
- **RewriteCond –** Specifies a condition under which a particular "RewriteRule" should be applied
- **RewriteRule –** Defines the actual rule for rewriting the URL. This rule is only executed if the preceding "RewriteCond" conditions are met

mod_rewrite flags

[R] – Issues an HTTP redirect. [R=302] for example
[P] – Handles requests via 'mod_proxy', making Apache a proxy server
[NC] – Makes the rule case-insensitive, matching regardless of capitalization
[L] – Stops processing further rules if this one matches

AWS Virtual Private Cloud

Secure, Isolated, and Customizable Networking

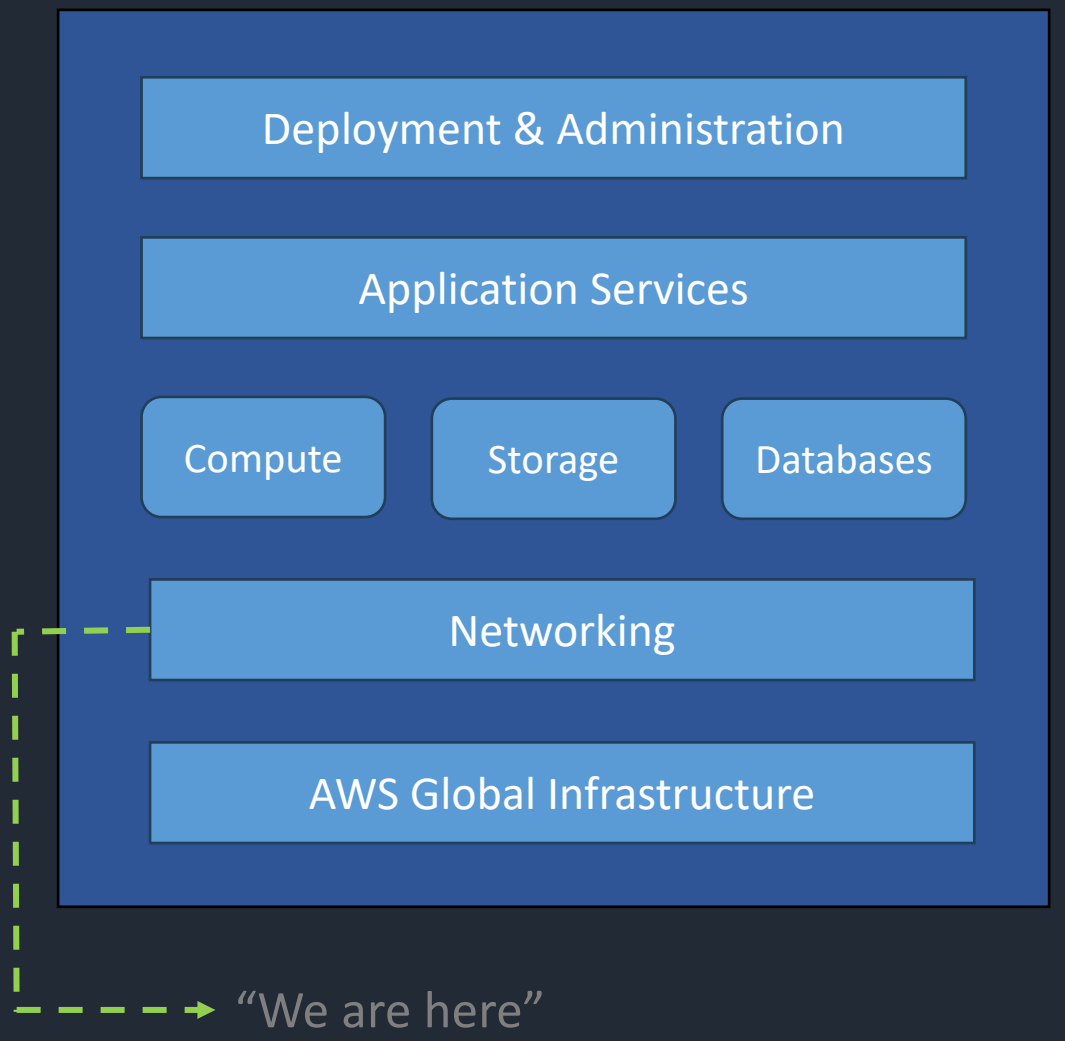
AWS VPC allows us to create a virtual network in AWS where we can launch AWS resources in a virtual network that we define

Our VPC will logically be isolated from other virtual networks in the AWS cloud

It allows to control IP address ranges, subnets, route tables, network gateways

It allows us to utilize Security Groups and Network ACLs to enforce inbound and outbound filtering at the instance and subnet level

We will be using VPC to create an internet and external network.
- Internal network to host C2 Server, RedELK logging server, Gophish, iRedMail and other instances that don't require inbound internet connection
- External network to host Redirector, Evilginx, and Payload Server which are the face of the infrastructure



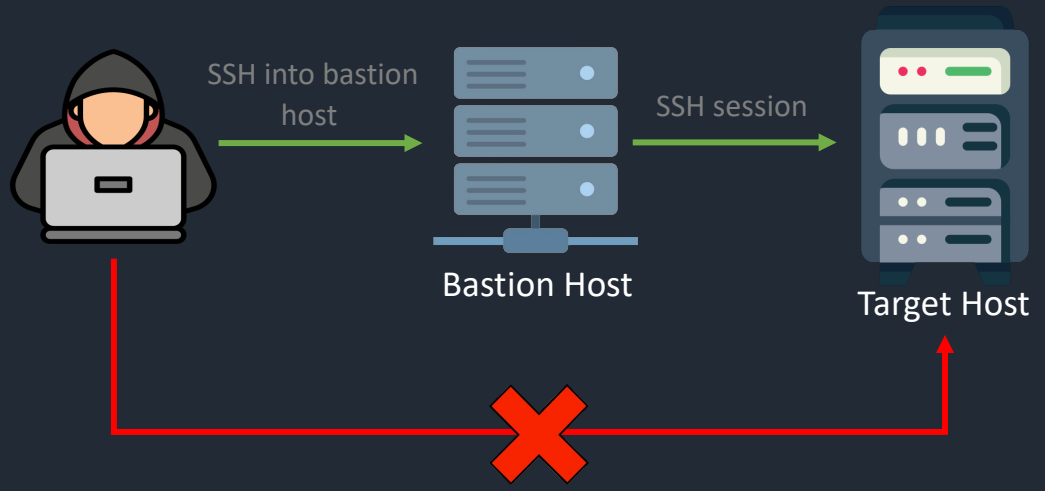
Bastion Host

Bastion Host, also known as “Jump Box”, serves as the entry point to a private network from an external network

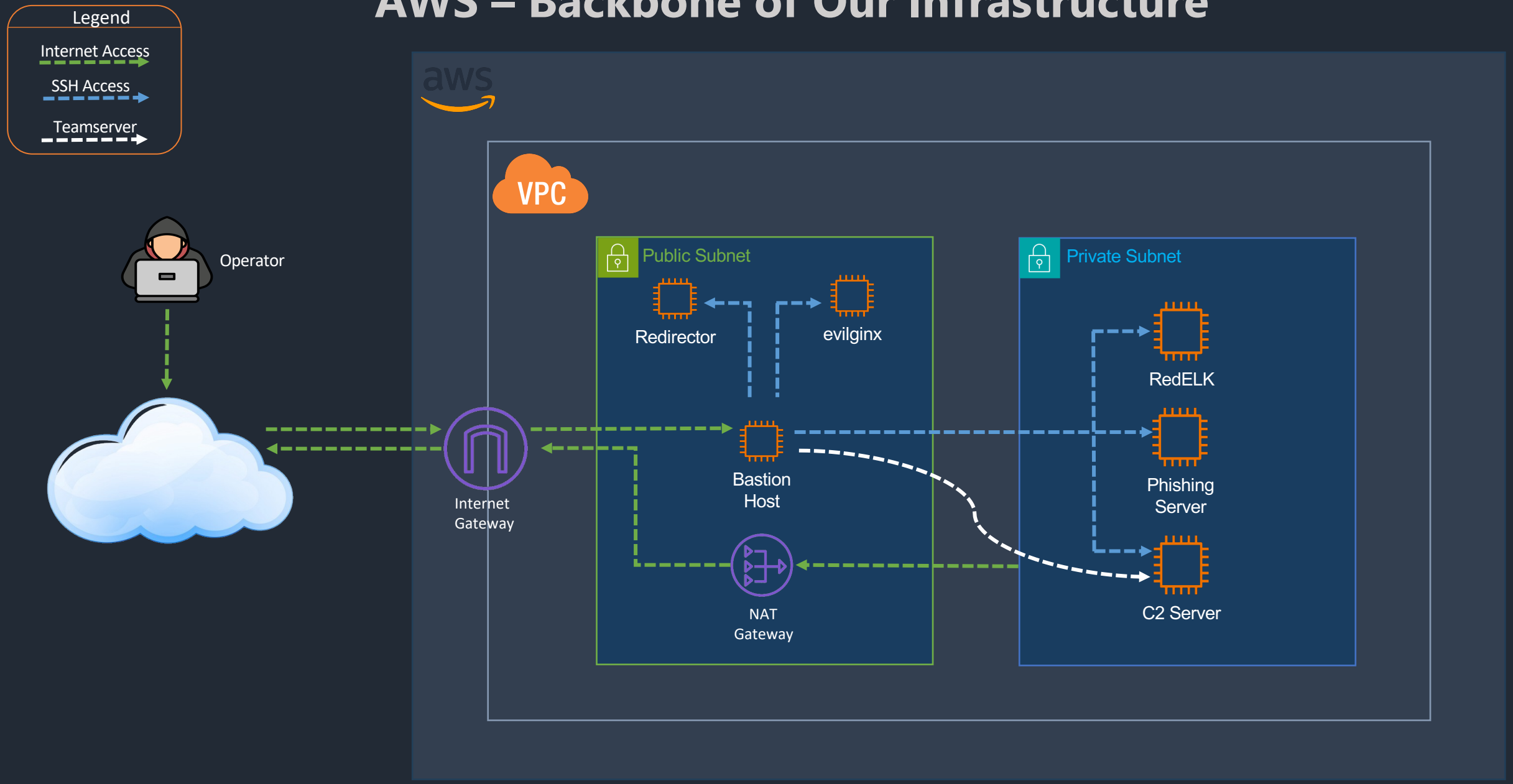
It provides a controlled and monitored entry point, limiting the exposure of other sensitive servers

We will setup our EC2 instances to accept SSH connections solely from the bastion host

Additionally, access to the C2 server’s Teamserver port will be exclusively through the bastion host which means to connect to C2 server, we’ll need to use port forwarding to channel the traffic via the bastion host



AWS – Backbone of Our Infrastructure



Terraform Crash Course

What?

Infrastructure as a Code tool developed by HashiCorp that allows you to describe your infrastructure using code.

This approach enables you to manage your infrastructure through version control systems, share configurations and reuse components

It can be used to manage low-level components like compute, storage and networking resources, as well as high-level components like DNS entries and SaaS features

Key Concepts

Providers – Plugins that allow Terraform to interact with cloud providers, SaaS providers and other APIs

Resource – Basic infrastructure components such as VM, Network Interface, Load Balancer etc

State – Terraform tracks the state of your managed infrastructure and configuration. This state is used to plan and make changes to your infrastructure

How?

1. Users define infrastructure in configuration files (*.tf) using HCL language which describes the desired state of your infrastructure
2. Before applying your configuration, you perform (terraform init) which downloads and installs necessary providers specified in your configuration file
3. You run (terraform plan) to compare your desired state to the actual state and show changes
4. Terraform tracks infrastructure in the state file “terraform.tfstate”

```
provider “aws” {
  access_key = “<accesskey>”
  secret_key = “<secretkey>”
  region     = “us-east-1”
}

resource "aws_instance" "example" {
  ami           = "ami-0c55b159cbfafe1f0"
  instance_type = "t2.micro"
}
```

Configuring Access Keys for Terraform

Terraform can utilize AWS credentials in several ways to authenticate against AWS account. Below are some common methods

Environment Variables

```
export AWS_ACCESS_KEY_ID="<accesskey>"
export AWS_SECRET_ACCESS_KEY="<asecretkey>"
```

AWS Credentials File inside ~/.aws/credentials

```
[default]
aws_access_key_id = <accesskey>
aws_secret_access_key = <secretkey>
```

AWS CLI Configuration – If awscli is installed then run `aws configure` to setup profile which will save the credentials to the above file

```
aws configure
```

Terraform Configuration

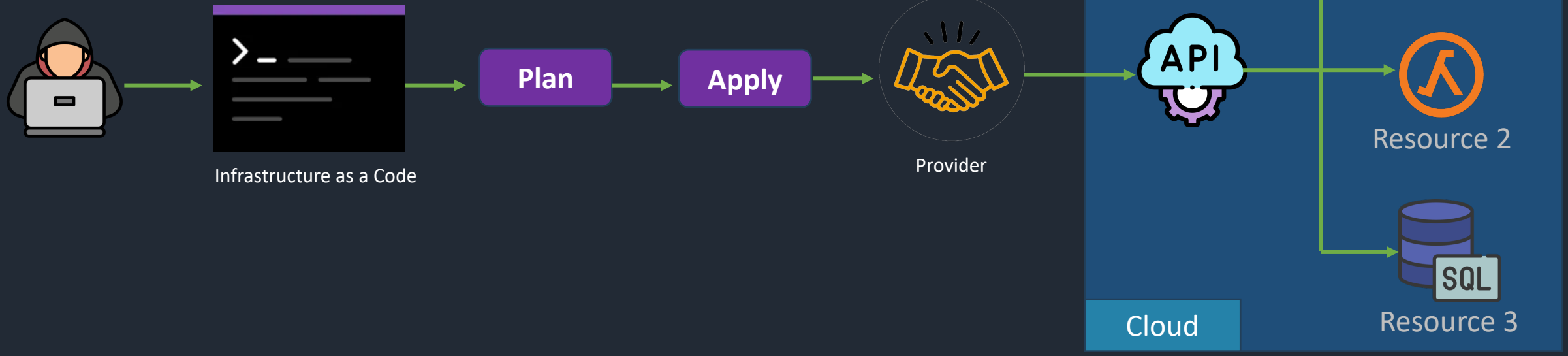
```
provider "aws" {
  access_key = "<accesskey>"
  secret_key = "<secretkey>"
  region     = "us-east-1"
}
```

Assume Role

```
provider "aws" {
  assume_role {
    role_arn     =
"arn:aws:iam::ACCOUNT_ID:role/ROLE_NAME"
    session_name = "session-name"
    external_id  = "external-id"
  }
}
```

Terraform orchestrates and maintains infrastructure elements on cloud platforms by leveraging their respective APIs

Providers act as the bridge for Terraform, facilitating communication with various cloud services through these APIs enabling infrastructure automation



Ansible Crash Course

What?

Open-source automation engine that automates software provisioning, configuration management and application deployment

It is agentless which makes it simple to automate tasks across infrastructure without needing to install specific agents on target systems

It uses SSH to connect to servers and run the configured tasks

Key Concepts

Playbooks – YAML files that describe the desired state of your system, tasks to be executed and order of execution

Inventory – Define the host or group of hosts on which commands, tasks and playbooks will run

Roles – Collection of playbooks to facilitate reuse and simplify playbook creation

Modules – Pre-built scripts used by Playbook to perform system tasks

How?

1. Users define the desired state of infrastructure in Playbooks (*.yml) using YAML, which outlines tasks to be performed on hosts
2. Before running your Playbooks, you set up an Inventory (inventory.ini) which lists all the hosts
3. You run (ansible-playbook <playbook_name>.yml) to execute the against the hosts specified in the Inventory
4. Ansible uses Facts, which are details about your system and environment, to decide how to execute the tasks and ensure the desired state

```
---
- name: Install Apache web server
  hosts: webservers
  become: yes
  tasks:
  - name: Install Apache
    apt:
      name: apache2
      state: latest
```

YAML 101

Fundamentals of YAML

- **Indentation –** Uses spaces (not tabs) for nesting elements
- **Key-Value Pairs –** “key: value” format for representing dictionary items
- **Lists –** A sequence of items denoted by a hyphen and space (‘- ‘)
- **Comments –** Use the ‘#’ symbol for comments

Ansible Playbook Anatomy

- **Tasks –** A list of actions to execute, each starting with ‘- name:’
- **Handlers –** Special tasks that run at the end of a playbook if notified
- **Roles –** Reusable playbook snippets that can be included in multiple playbooks

Sample Ansible YAML Playbook

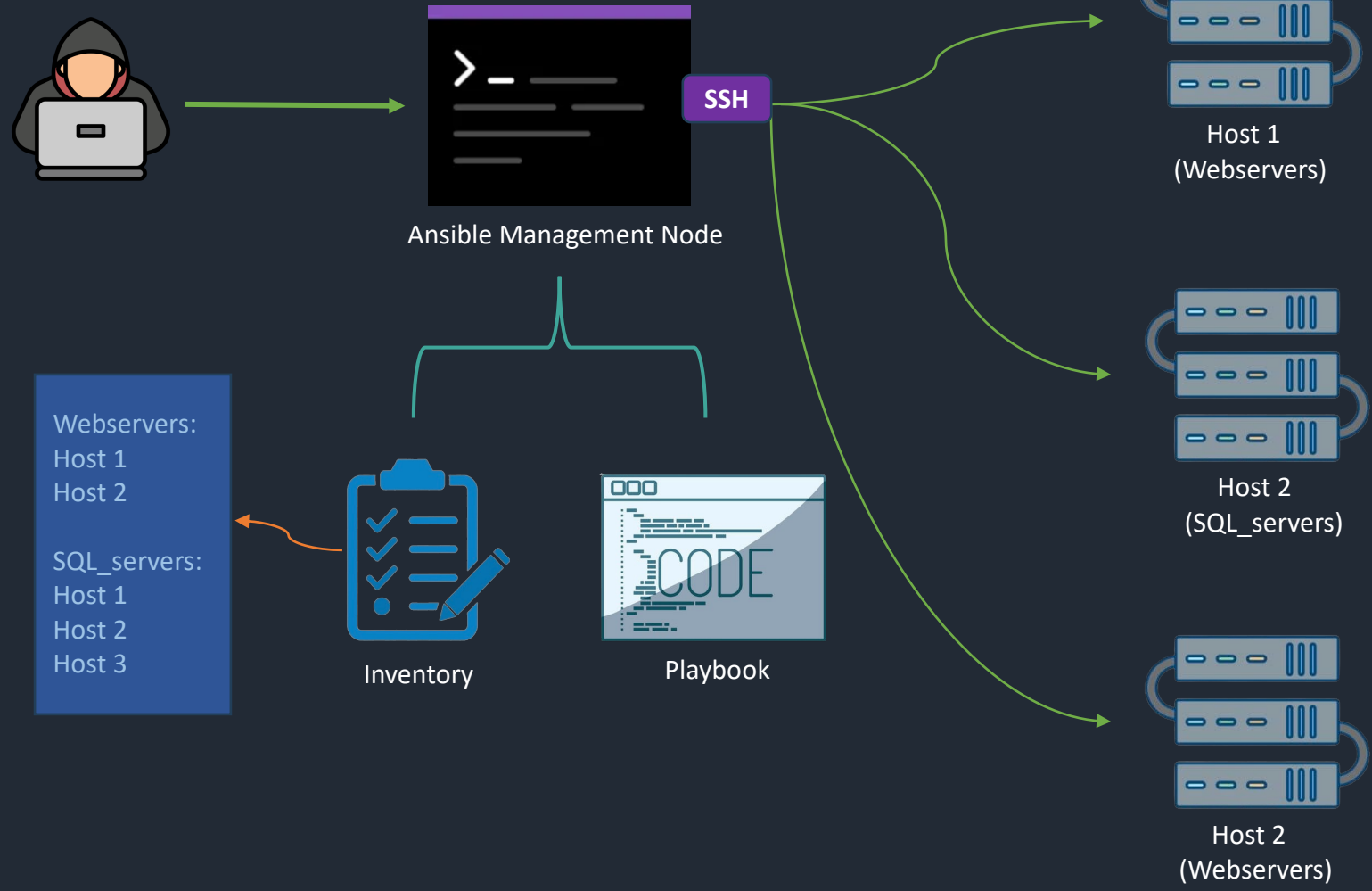
```
tasks:
  - name: Template configuration file
    template:
      src: template.j2
      dest: /etc/apache_http.conf
    notify:
      - restart the apache service

handlers:
  - name: restart apache service
    service:
      name: apache2
      state: restarted
```

In Ansible, a handler is a special kind of task that only runs when it is notified by another task.

It runs everything over SSH, which is a secure way to access nodes remotely

Ansible automates tasks by connecting to nodes defined in the inventory and executes tasks defined inside the Ansible Playbook

Once Ansible completes the tasks, it cleans up after itself, leaving no extra software or agents behind



Terraform vs Ansible

	Terraform	Ansible
Purpose	Terraform is primarily used for provisioning and managing the infrastructure like creating servers, databases, networks etc.	Ansible is more focused on configuring and orchestrating the software within existing systems, like installing and updating programs on servers
State Management	It keeps a state file to track the infrastructure it manages	It doesn't track state, so it's better for managing things that change over time
Immutability	Terraform is immutable, meaning it will replace the existing infrastructure when a configuration is updated	Ansible is mutable, it applies changes in place and can manage the ongoing state of the system
Declarative	You describe the desired end-state of the infrastructure and terraform will take care of it	You describe the end-state but also the steps to reach that state

Terraform is more ideal for setting and maintaining the actual infrastructure on cloud platforms. We will use terraform to provision various cloud resources such as EC2 instances, CDN Providers, MailGun, Namecheap service etc.

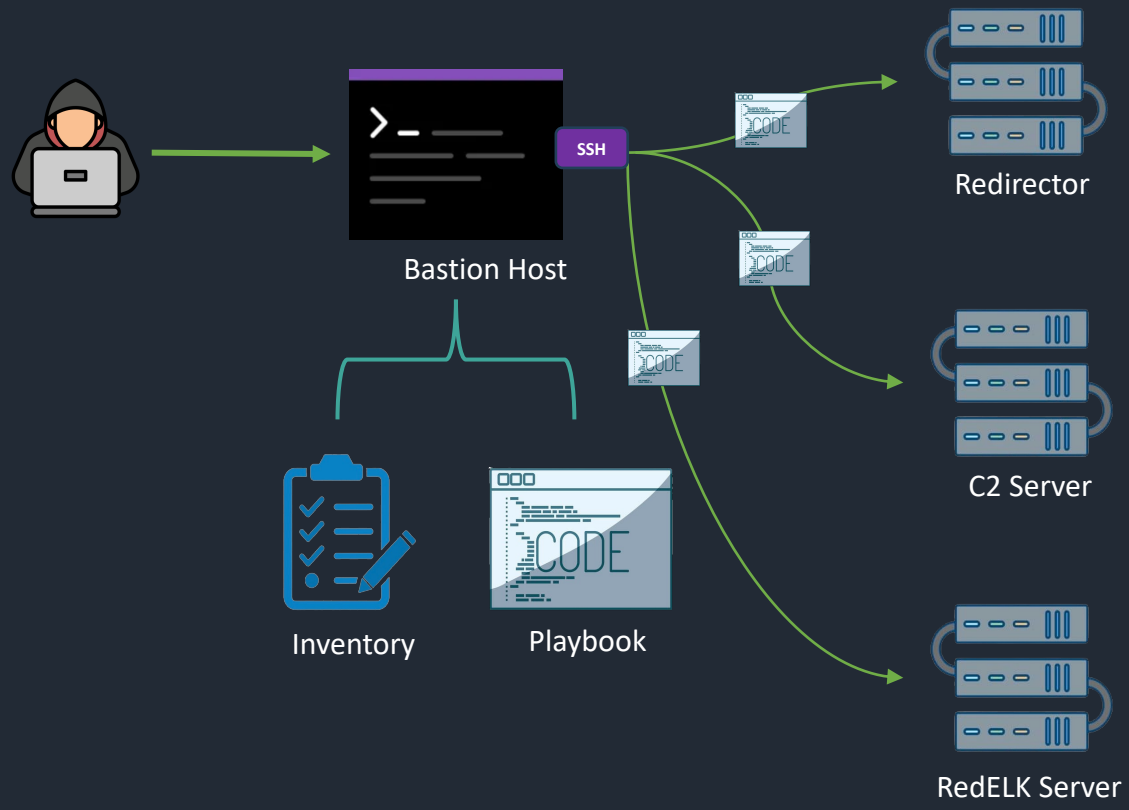
Ansible excels at configuring and maintaining the software and systems on that provisioned infrastructure. We will be using Ansible for hardening our provisioned infrastructure, setting up Redirector services, RedELK and other services

We will be installing Ansible on Bastion Host

Bastion host will be utilized for setting up each component via Ansible

For instance, following is the sample code for setting up Teamserver via bastion host

```
resource "null_resource" "setup_teamserver" {
    depends_on = [aws_instance.teamserver]

    count = var.install_redelk ? 1 : 0 # only run if redelk=true

    connection {
        type = "ssh"
        user = var.ssh_user
        host = var.bastionhostpublicip
       private_key = var.private_key
    }

    provisioner "remote-exec" {
        inline = [
          "ansible-playbook -i inventory.ini teamserver.yml"
        ]
    }
}
```



Gophish

A robust email service that provides reliable delivery for bulk emails

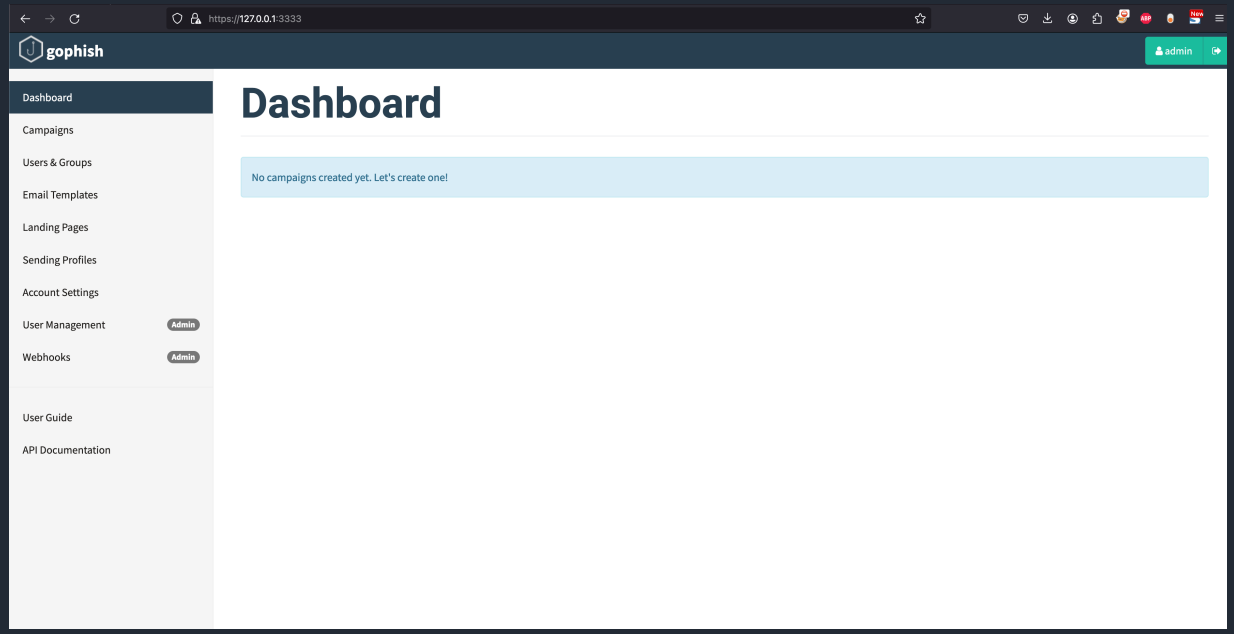
GoPhish requires SMTP credentials for email delivery

Integrating GoPhish with established SMTP servers like Mailgun bypasses the need to build and maintain our own, ensuring higher deliverability rates.

Mailgun have invested heavily in their sending reputation Their trusted status increases the likelihood of landing in inboxes, not spam folders.

Easy integration with existing tools and frameworks via API

Now can be integrated with evilginx ☺



Enhancing Email Delivery with MailGun

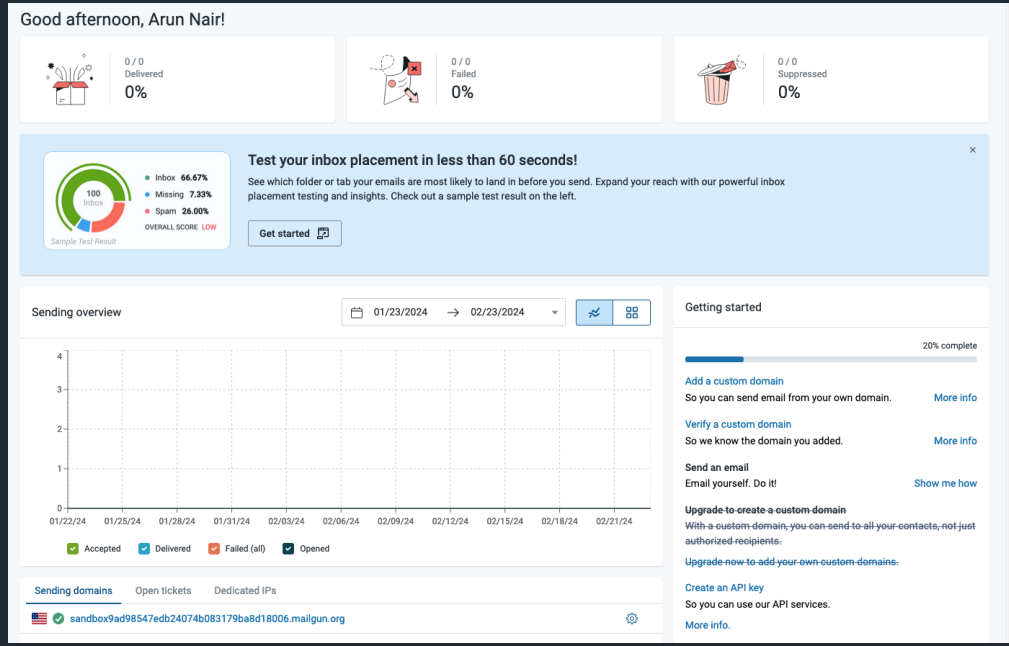
A robust email service that provides reliable delivery for bulk emails

GoPhish requires SMTP credentials for email delivery

Integrating GoPhish with established SMTP servers like Mailgun bypasses the need to build and maintain our own, ensuring higher deliverability rates.

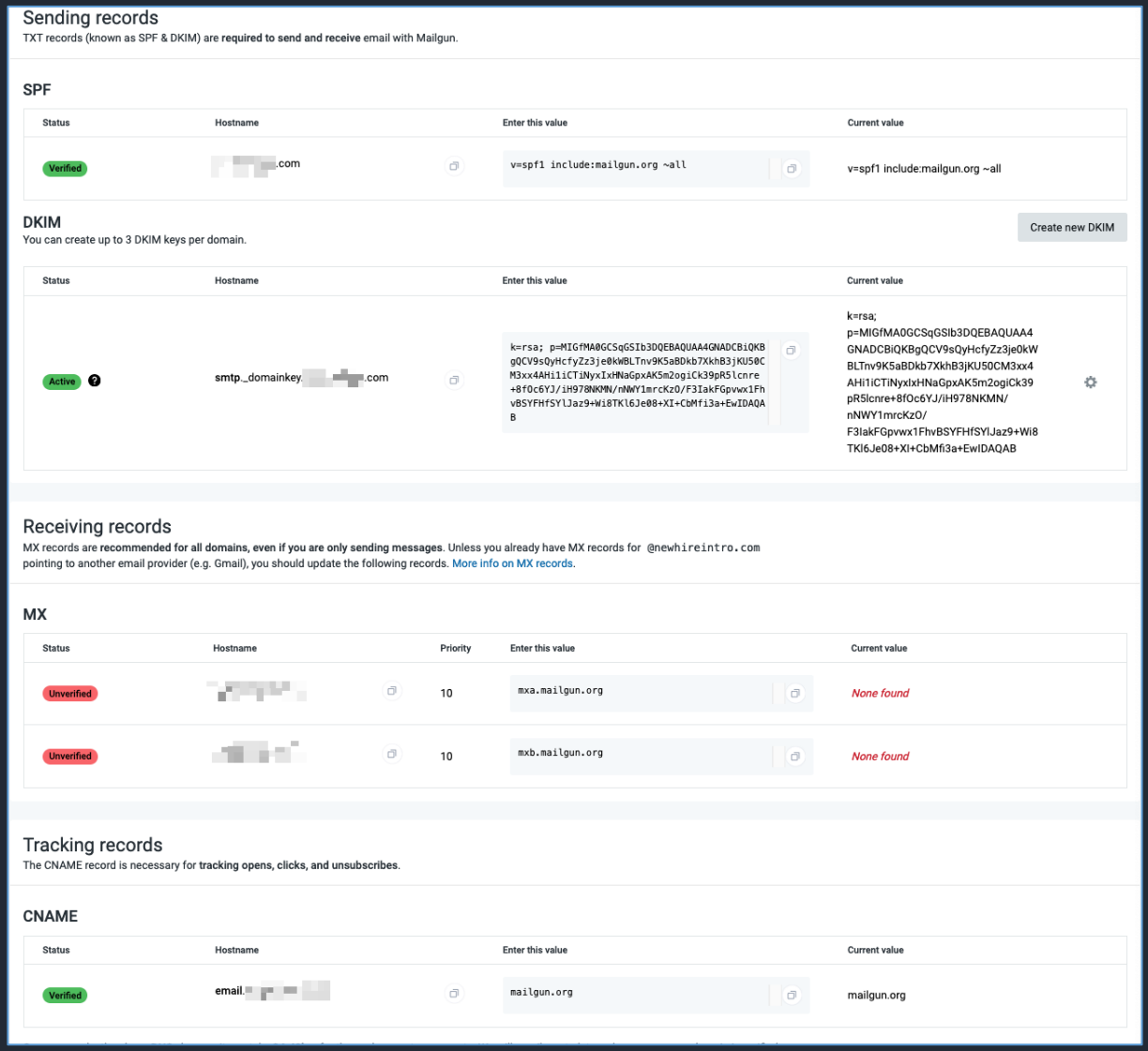
Mailgun have invested heavily in their sending reputation Their trusted status increases the likelihood of landing in inboxes, not spam folders.

Easy integration with existing tools and frameworks via API



Automating MailGun Setup

```
terraform {
  required_providers {
    mailgun = {
      source = "wgebis/mailgun"
      version = "0.7.4"
    }
  }
}

provider "mailgun" {
    api_key = ”<API-KEY>"
}


# Create a new Mailgun domain
resource "mailgun_domain" "default" {
    count = length(var.mailgun_domain_name)
    name = "${var.mailgun_domain_name[count.index]}"
    region = var.mailgun_region
    spam_action = "disabled"
    dkim_key_size = 1024
}

. . . # REST CODE FOR FILLING IN SPF, DKIM, MX AND CNAME RECORD
```



RedELK – Downloading on Bastion

redelkconfig.cnf

```
[req]
distinguished_name = req_distinguished_name
x509_extensions = v3_req
prompt = no

[req_distinguished_name]
emailAddress = root@localdomain

[v3_ca]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
basicConstraints = CA:TRUE

[v3_req]
keyUsage = keyEncipherment, dataEncipherment
extendedKeyUsage = serverAuth
subjectAltName = @alt_names

[alt_names]
# Enter the valid IP or DNS where the teamservers and
redirectors can reach your ELK server in the fields below. If
not done correclty, Logstash on your ELK server will crash
with cryptic errors.
IP.1 = 127.0.0.1
IP.2 = 10.0.2.103 # we are using static internal IP for our
redelk server
```

```
---
- name: Download RedELK with Ansible
  hosts: localhost # gonna run on bastion host
  connection: local
  become: yes # Use sudo for tasks that require it
  vars:
    redelk_repo: "https://github.com/outflanknl/RedELK"
    redelk_install_dir: "/home/ubuntu/RedELK"
    config_file_source: "/tmp/redelkconfig.cnf"
    config_file_dest: "{{ redelk_install_dir }}/certs/config.cnf"

tasks:
- name: Clone RedELK repository
  ansible.builtin.git:
    repo: "{{ redelk_repo }}"
    dest: "{{ redelk_install_dir }}"
    clone: yes
    update: yes

- name: Move configuration file to RedELK directory
  ansible.builtin.copy:
    src: "{{ config_file_source }}"
    dest: "{{ config_file_dest }}"
    remote_src: no

- name: Run initial setup script
  ansible.builtin.shell:
    cmd: "./initial-setup.sh certs/config.cnf"
    chdir: "{{ redelk_install_dir }}"
    executable: /bin/bash
```

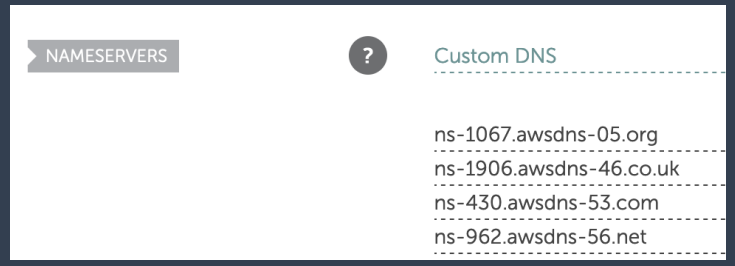
Namecheap to Route53

Given our infrastructure resides on AWS, so for efficiency, it’s better to centralize domain management on AWS too

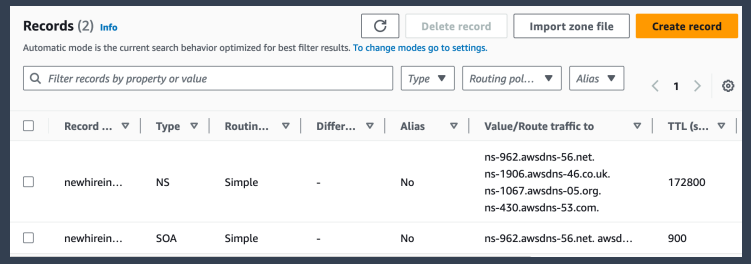
“aws_route53_zone” resource is used to create a new DNS zone in AWS Route53

“namecheap_domain_records” resource is used to update the nameservers for our domain to the ones provided by AWS Route 53

After executing the terraform script, AWS nameservers will be configured in Namecheap, as shown below



On AWS, it will look something like the following




```
terraform {
  required_providers {
    namecheap = {
      source = "namecheap/namecheap"
      version = "2.1.0"
    }
  }
}

provider "namecheap" {
    user_name = ”<namecheap_username>"
    api_user = ”<namecheap_apiuser>"
    api_key = ”<namecheap_api_key>"
}

resource "aws_route53_zone" "zone" {
    for_each = toset(var.domains)
    name = each.value
    comment = "moving domains"
    tags = {
        Name = "route53_zone"
    }
}

resource "namecheap_domain_records" "mydns" {
    for_each = aws_route53_zone.zone
    domain = each.value.name
    mode = "OVERWRITE"
    nameservers = each.value.name_servers
}
```

Azure CDN Redirector

Why Azure Front Door CDN?

- Offers built-in DDoS protection and application layer security
- Leverages Microsoft's global network for low-latency content delivery
- Ensures high availability with smart health probes and instant global failover

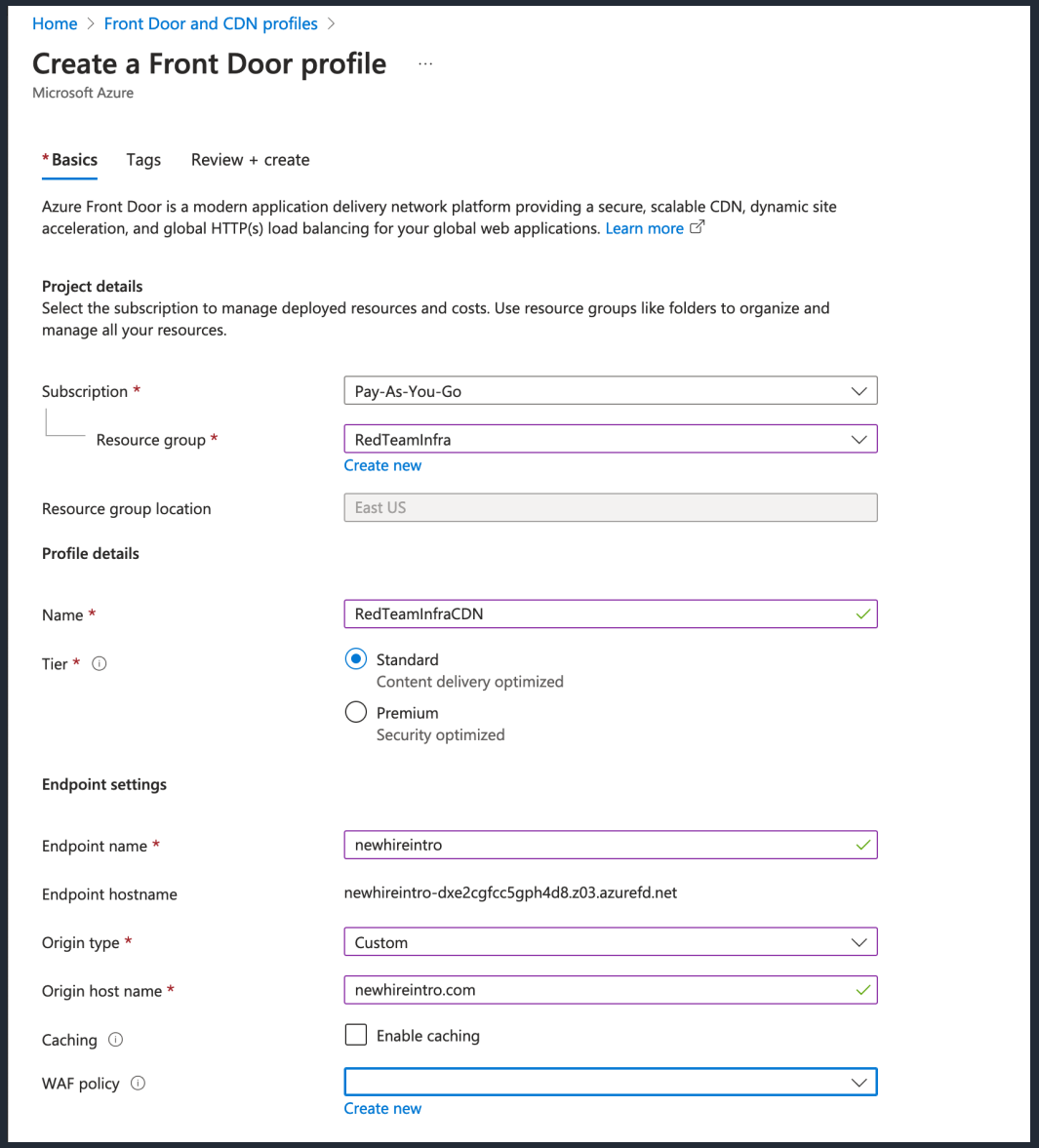
Azure Front Door Configuration

- Define a unique name for the CDN endpoint, which serves as the client-facing URL
- Configure to allow HTTPS traffic from clients
- Specify the IP address or domain name where the traffic will be forwarded if the condition is matched
- Disable caching to prevent payload failures

For more information on setting up Azure Front Door CDN, refer to the following articles

- https://medium.com/r3d-buck3t/red-teaming-in-cloud-leverage-azure-frontdoor-cdn-for-c2-redirectors-79dd9ca98178

- https://bigb0ss.medium.com/redteam-c2-redirector-domain-fronting-setup-azure-adbedbd28305



AzureCDN Redirector Terraform

1

```
~                                                          09:57:06 PM
az login
A web browser has been opened at https://login.microsoftonline.com/organizations/
orize. Please continue the login in the web browser. If no web browser is availab
 browser fails to open, use device code flow with `az login --use-device-code`.
The following tenants don't contain accessible subscriptions. Use 'az login --all
ons' to have tenant level access.
[redacted]
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "[redacted]",
    "id": "[redacted]",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Pay-As-You-Go",
    "state": "Enabled",
    "tenantId": "[redacted]",
    "user": {
      "name": "arun.nair[redacted]com",
      "type": "user"
    }
  }
]
```

2

```
terraform {
    required_providers {
      azurerm = {
        source = "hashicorp/azurerm"
        version = "=3.0.0"
      }
    }
}

provider "azurerm" {
    features {}
}
```

3

```
resource "azurerm_cdn_endpoint" "cdn" {
  name = var.cdn_endpoint_name
  profile_name = azurerm_cdn_profile.cdn_profile.name
  location = azurerm_resource_group.cdn_resource_group.location
  resource_group_name = azurerm_resource_group.cdn_resource_group.name
  is_compression_enabled = false
  optimization_type = "GeneralWebDelivery"
  querystring_caching_behaviour = "BypassCaching"

  origin {
    name = “<domain_name>”
    host_name = “<domain_name>”
    http_port = 80
    https_port = 443
  }

  delivery_rule {
    name = "RedirectRule"
    order = 1
    request_header_condition {
      selector = "X-ASPNET-VERSION"
      operator = "Equal"
      match_values = ["1.7”]
      negate_condition = true
    }
    url_redirect_action {
      redirect_type = "Found"
      protocol = "Https"
      hostname = “www.google.com”
      path = "/"
    }
  }
}
```

Implementing SSL to HTTP Redirector

As traffic in HTTP is clear text, it's best to avoid it as much as possible

Let's Encrypt self-signed certificates can be used to enable HTTPS

Certbot can also create self-signed certificates by obtaining Let's Encrypt certificate in the backend

Ensure the "A" Record for your domain points to the IP address where Certbot is being executed

```
certbot certonly -d <redirector_domain> --webroot
-w /var/www/html/<redirector_domain> --register-
unsafely-without-email
```

Certificate files will be placed in "/etc/letsencrypt/live/<redirector_domain>/

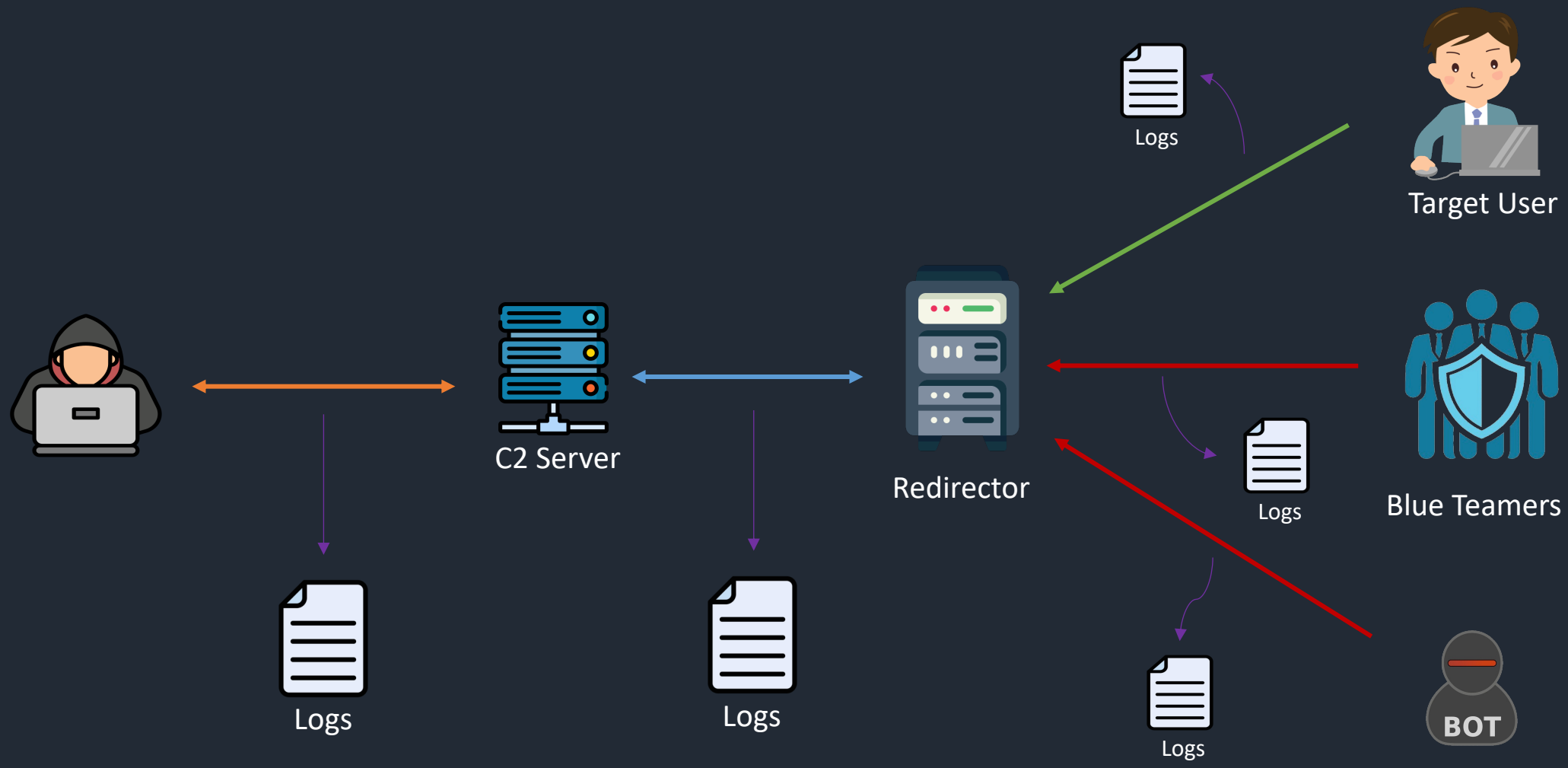
Modify the Apache configuration to point to the new certificate and key

Ensure mod_ssl is enabled ("sudo a2enmod ssl")

httpsredirs.conf

```
<VirtualHost *:443>

ServerName <redirector_domain>
DocumentRoot /var/www/html/<redirector_domain>

ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined

SSLEngine On
SSLVerifyClient none
SSLProxyEngine On
SSLProxyCheckPeerName off
SSLProxyVerify none
SSLProxyCheckPeerCN off
SSLProxyCheckPeerName off
SSLProxyCheckPeerExpire off

SSLCertificateFile /etc/letsencrypt/live/DOMAIN/cert.pem
SSLCertificateKeyFile /etc/letsencrypt/live/DOMAIN/privkey.pem
SSLCertificateChainFile /etc/letsencrypt/live/DOMAIN/fullchain.pem
#UNCOMMENT_FOR_COUNTRY_REDIR Include /etc/apache2/modmaxmind.conf

RewriteEngine On
# If the request hits /rofl/ then redirect to C2
RewriteCond %{REQUEST_URI} ^/rofl/(\S+)$
RewriteRule ^.*$ https://C2IP:443%{REQUEST_URI} [P]
ProxyPassReverse ^ https://C2IP:443
# If the above rules don't match, send to Google.
RewriteRule ^.*$ http://www.google.com/? [L,R=302]

</VirtualHost>
```

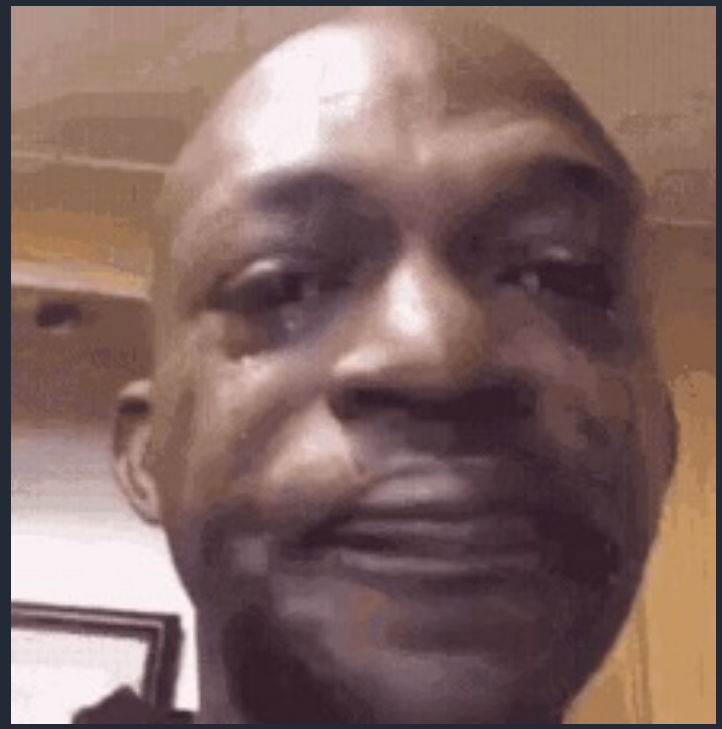
Logs Everywhere



The Logs Which Are Everywhere

```
ubuntu@httpredir1:~$ cat /var/log/apache2/access-redelk.log
[02/Apr/2024:16:03:52 +0000] - apache[2946]: frontend:-/10.0.1.222:80 backend:decoy-amazon client:101.0.63.209:1765
1 xforwardedfor:- headers:{Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/124.0|34.
207.82.92|-|-|-|-|-|} statuscode:302 request:GET / HTTP/1.1
[02/Apr/2024:16:04:01 +0000] - apache[2947]: frontend:-/10.0.1.222:80 backend:decoy-amazon client:101.0.63.209:1756
6 xforwardedfor:- headers:{Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/124.0|34.
207.82.92|-|-|-|-|-|} statuscode:302 request:GET /google/abc HTTP/1.1
[02/Apr/2024:16:04:12 +0000] - apache[2946]: frontend:-/10.0.1.222:80 backend:decoy-amazon client:101.0.63.209:1763
7 xforwardedfor:- headers:{Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/124.0|34.
207.82.92|-|-|-|-|-|} statuscode:302 request:GET /google/aslsl HTTP/1.1
[02/Apr/2024:16:10:02 +0000] HOSTNAME apache[2946]: frontend:www-http/10.0.1.222:80 backend:c2 client:101.0.63.209:
17726 xforwardedfor:- headers:{Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/124.0
|34.207.82.92|-|-|-|-|-|} statuscode:503 request:GET /ramukaka/abc HTTP/1.1
[02/Apr/2024:16:10:03 +0000] - apache[2947]: frontend:-/10.0.1.222:80 backend:decoy-amazon client:101.0.63.209:1753
1 xforwardedfor:- headers:{Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/124.0|34.
207.82.92|-|-|-|-|-|} statuscode:302 request:GET /favicon.ico HTTP/1.1
[02/Apr/2024:16:12:07 +0000] - apache[2947]: frontend:-/10.0.1.222:80 backend:decoy-amazon client:82.32.97.79:48794
 xforwardedfor:- headers:{-|34.207.82.92|-|-|-|-|-|} statuscode:302 request:GET / HTTP/1.1
ubuntu@httpredir1:~$
```

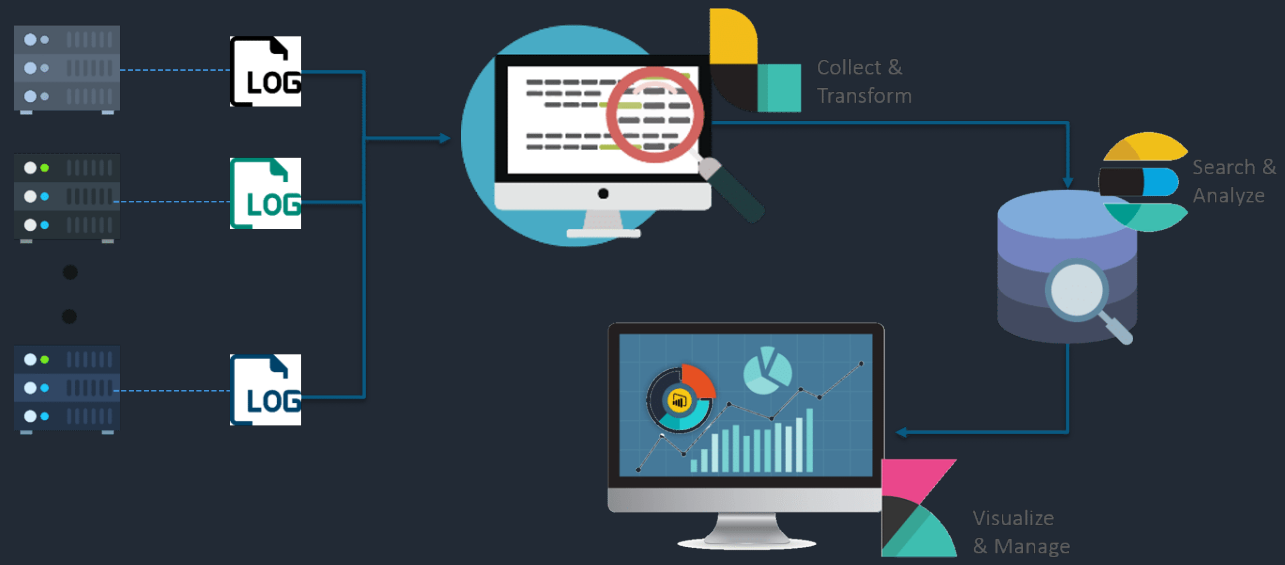
```
06/19 14:04:54 [metadata] beacon_28131 -> 172.16.20.80; computer: GRANITE; user: raffi; pid: 5332; version: 6.1
06/19 14:04:53 [output]
established link to parent beacon: 172.16.20.80

06/19 14:04:56 [input] <neo3> ps
06/19 14:04:56 [task] Tasked beacon to list processes
06/19 14:04:57 [checkin] host called home, sent: 12 bytes
06/19 14:04:57 [output]
[System Process]        0       0
System  0       4
smss.exe        4       312
csrss.exe       396     420
wininit.exe     396     516
csrss.exe       508     528
winlogon.exe    508     592
services.exe    516     616
lsass.exe       516     648
lsm.exe 516     656
svchost.exe     616     764
```



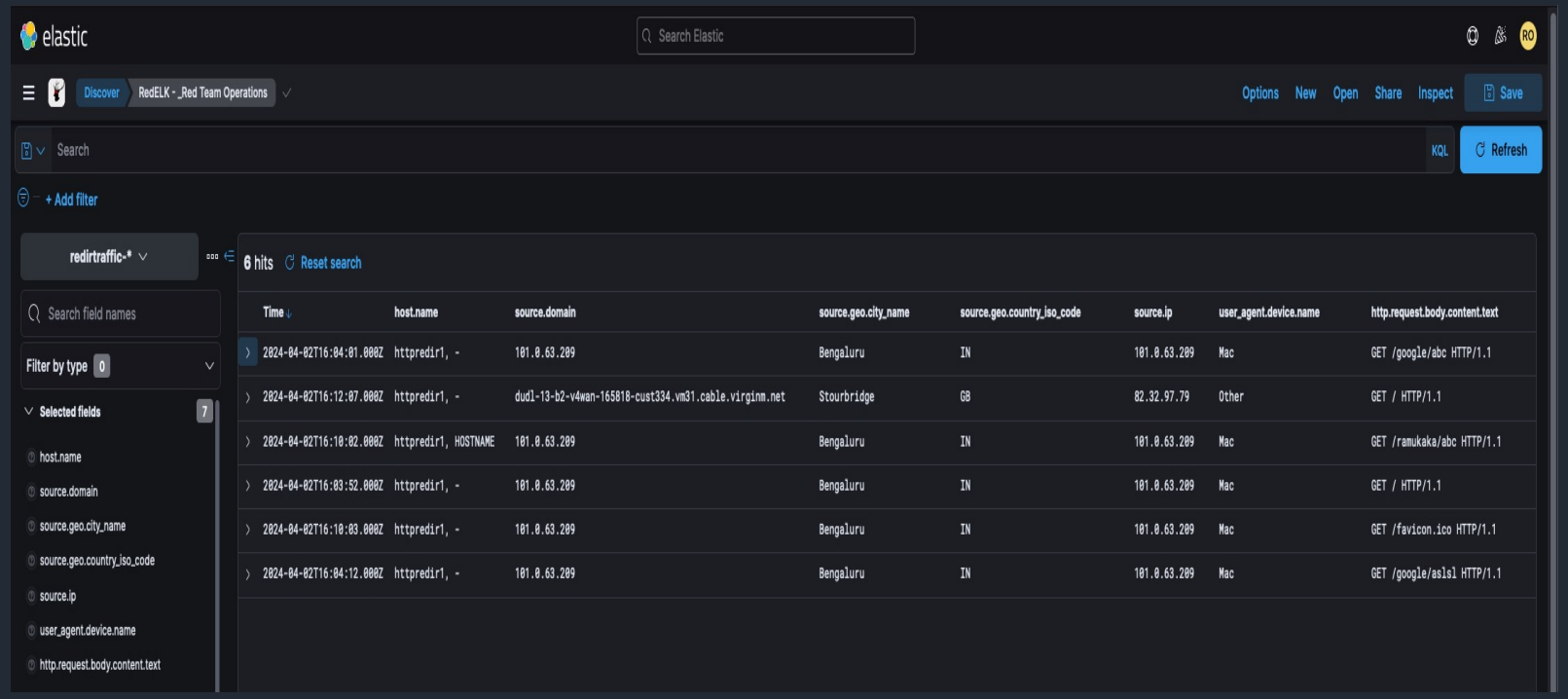
ELK Stack

ELK Stack is a collection three open-source products; Elasticsearch, Logstash and Kibana

Elasticsearch: Store, Search and Analyse
Logstash: Collect logs and events data, Parse and Transform
Kibana: Explore, Visualize and Share
Beats: Data Shipper



Logging with RedELK



Thank You

Any Questions or Feedback?