

ExploderBot: Fast Flux Botnet Infrastructure Guide

Jeremy Hastings
jth0024@uah.edu
256.738.5115
Principal Researcher

Kevin Jernigan
kmj0019@uah.edu
931.908.9832
Principal Researcher

Cristina Ramos
cir0002@uah.edu
915.667.2020
Principal Researcher

Stanley Chapman
sc0149@uah.edu
256.797.3838
Principal Researcher

Table of Contents

Host Requirements	3
Clone ExploderBot Repository	3
Create Python Virtual Environment	3
Install Python Packages	3
Available Ansible Playbooks	4
AWS – Create Identity Access Management (IAM) Users	5
AWS – Maintaining User Access Keys	8
AWS – Create SSH Public Key Pair	8
Create an Ansible Vault	10
Run an Ansible Playbook	10

Host Requirements

- Operating System (OS):
 - Linux Derivative – Successfully tested with RHEL/CENTOS and Debian-based Distributions
- Packages
 - git
 - aws-cli
 - ansible
 - python3 – Confirmed compatibility with Python 3.6 and above
 - python3-pip
 - virtualenv
- Infrastructure Requirements
 - An Amazon Web Services (AWS) Account
 - AWS – Virtual Private Cloud
 - AWS – Subnet
 - AWS – Internet Gateway

Clone ExploderBot Repository

All necessary source-code to replicate the ExploderBot Fast Flux Infrastructure is hosted via a Github repository. Utilize the git command-line interface (cli) to clone the repository:

```
# git clone https://github.com/cristisabela/dnstool
```

Create Python Virtual Environment

Utilizing the virtualenv utility, create and activate a virtual environment:

```
# virtualenv [virtualenv name]
# source [virtualenv]/bin/activate
```

Install Python Packages

Utilize pip3 to install all necessary packages list in the “requirements.txt” file:

```
(virtualenv)# pip3 install -r requirements.txt
```

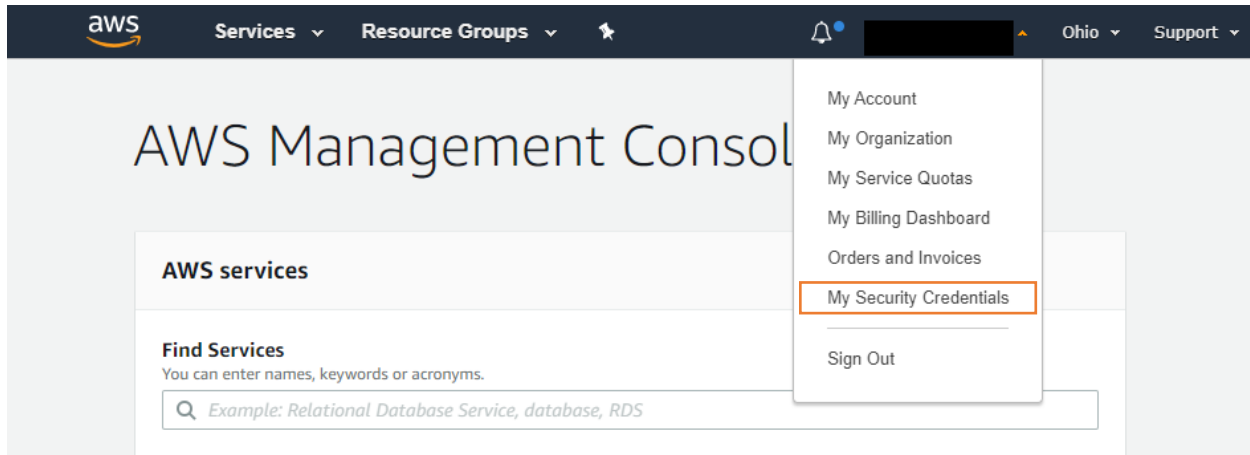
Available Ansible Playbooks

The ExploderBot Fast Flux Infrastructure repository include multiple ansible playbooks designed to automate the deployment of key pieces of infrastructure:

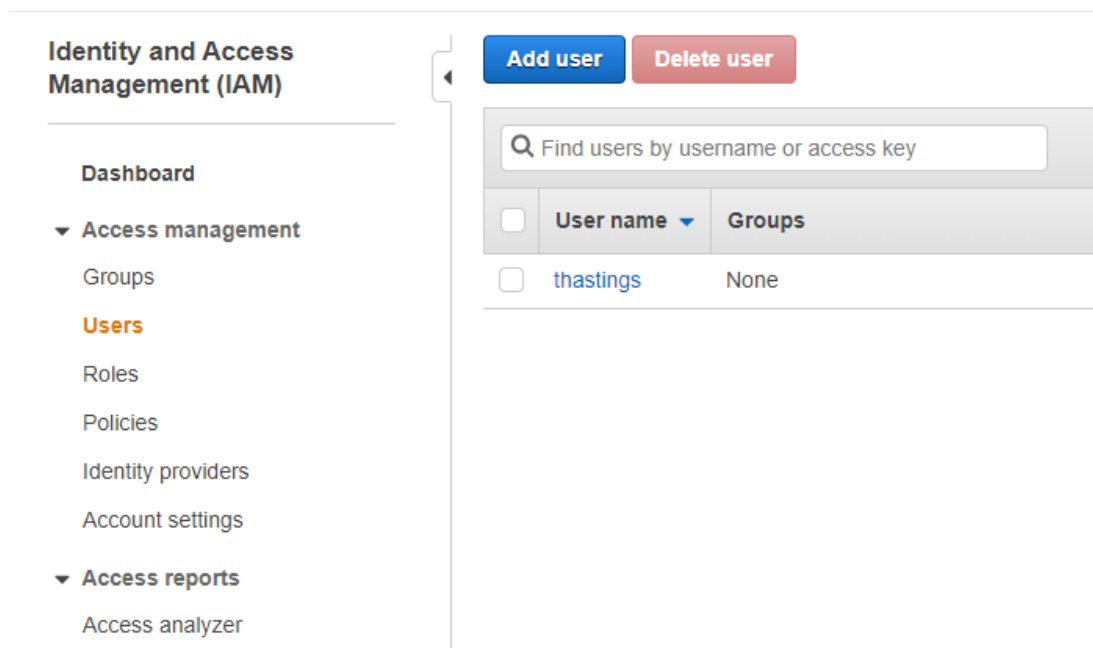
- **DNS Management & Analysis Playbook – dns_server_playbook.yml**
 - The following playbook provides the following capabilities:
 - Functioning DNS Server With Preconfigured Forward Zone: This forward zone currently contains records for an artificial domain: exploderbot-uah.com; however, this may be modified as needed.
 - Automated DNS Data Collection Service: A service configured to execute on startup collects all DNS queries and responses from the DNS server and stores them in a packet capture (PCAP) format. Additionally, the service will restart after running for 24 hours. PCAP files may be located within the /tmp directory, writted as: "/tmp/\${DATE}.pcap.
 - Automated AWS EC2 Elastic IP Utility: A Python script that utilizes AWS APIs to interact with a specified EC2 instance. Leverages Elastic IP services to associate a new public IP Address to a specified EC2 instance at a user-defined interval. This concept provides the fast-flux emulation capability
- **C2 Server Playbook – c2_playbook.yml**
 - The following playbook provides multiple utilities to emulate malicious activity:
 - DNS Tunneling/C2
 - TCP/HTTP(s) C2
 - Malware Hosting

AWS – Create Identity Access Management (IAM) Users

AWS IAM enables secure access to AWS resources. Using IAM, multiple users may interact with a single subscription and can be managed through groups and permissions. Create an IAM user by accessing the IAM Dashboard:



- The IAM Dashboard is most easily access by clicking the account’s username, located in the AWS navigation bar, and selecting “My Security Credentials”



- Select the “Users” option located under the “Access management” tab within the IAM Dashboard’s Sidebar and then select “Add user”

Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name*

[+ Add another user](#)

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)


- Access type*** ☒ **Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- ☒ **AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.


Console password* ☒ Autogenerated password
☐ Custom password


Require password reset ☒ Users must create a new password at next sign-in
Users automatically get the [IAMUserChangePassword](#) policy to allow them to change their own password.

- Fill out the form with the appropriate username and password settings. Ensure that the “Programmatic access” and “AWS Management Console access” check boxes are both selected and select “Permissions” to proceed.

▼ Set permissions




 Add user to group

 Copy permissions from existing user

 Attach existing policies directly

Create policy

Filter policies ▼

	Policy name ▼	Type	Used as
<input checked="" type="checkbox"/>	 AdministratorAccess	Job function	Permissions policy (1)
<input type="checkbox"/>	 AlexaForBusinessDeviceSetup	AWS managed	None
<input type="checkbox"/>	 AlexaForBusinessFullAccess	AWS managed	None

- Set the desired permissions the created users by either creating a user group, or attaching existing AWS policies directly to a user.
 - Note: Standard Users of this infrastructure will only need the “AmazonEC2FullAccess” policy selected to replicate the Fast Flux infrastructure.
- After selecting policies, proceed to “Tags” and provide IAM tags as needed. Finally, proceed to “Review” and finish by clicking “Create user”

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	test
AWS access type	Programmatic access and AWS Management Console access
Console password type	Autogenerated
Require password reset	Yes
Permissions boundary	Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	AdministratorAccess
Managed policy	IAMUserChangePassword

Tags


No tags were added.


AWS – Maintaining User Access Keys

✓ **Success**

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: [REDACTED] signin.aws.amazon.com/console

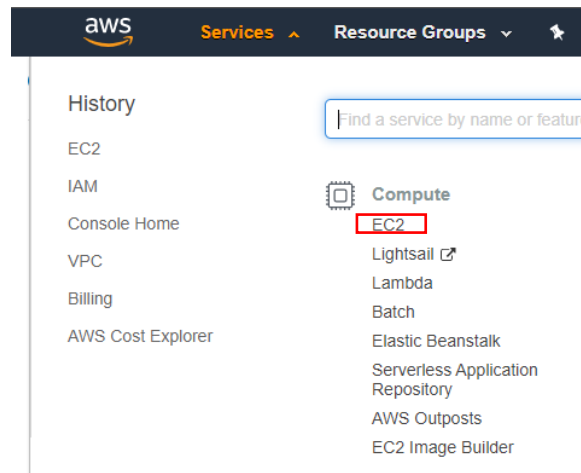
 Download .csv

	User	Access key ID	Secret access key	Password	Email login instructions
▶	✓ test	AKIAYETHRWYJGEOZE756	***** Show	***** Show	Send email 

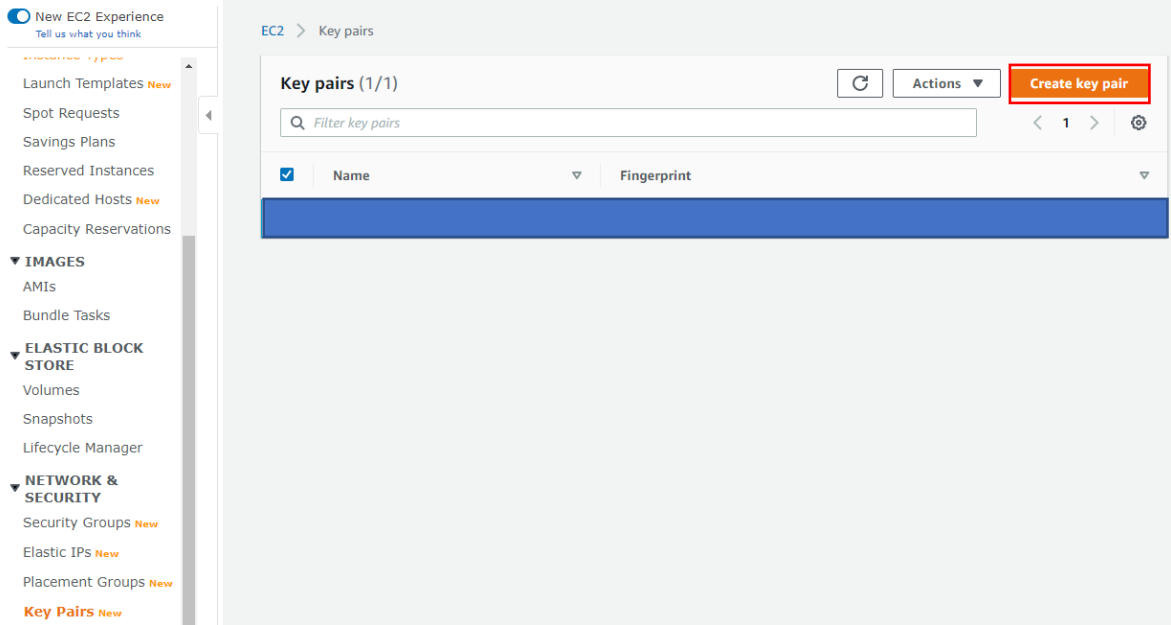
- After successfully creating a new user(s), a unique Access key ID and Secret access key will be generated. This is only time that these credentials will be accessible to review. Ensure that these are securely stored and disseminated to the appropriate user.

AWS – Create SSH Public Key Pair

AWS Elastic Compute Cloud (EC2) instances utilize Secure Shell (SSH) public keys for authentication. During EC2 creation, an SSH key is associated with an instance. Create an SSH Key for the user:



- Select the “Services” tab from the AWS navigation bar and click “EC2” under the “Compute” tab.



- Under the EC2 Dashboard, select “Key Pairs” under the “Network and Security” tab.

Key pair

A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name

The name can be up to 255 characters long. Valid characters include `_`, `-`, `a-z`, `A-Z`, and `0-9`.

File format

☒ pem
For use with OpenSSH

☐ ppk
For use with PuTTY

Cancel **Create key pair**

- Enter a key pair name and ensure that the “pem” File format is selected. Click “Create key pair” to finish. Download and store the public key.
 - Note – This will be the only time in which the newly create public key will be available for download.

```
- hosts: local
  connection: local
  gather_facts: False
  vars:
    instance_type: t2.micro
    security_group: dns_sg
    image: ami-0520e698dd500b1d1
    keypair: VPN
```

- To utilize the new keypair, edit an Ansible Playbook and modify the keypair variable with the new key pair name.

Create an Ansible Vault

Ansible Vault is a feature of ansible that provides encrypted storages of sensitive passwords or keys, rather than as plaintext values in a playbook or role. These vault files can then be distributed or placed in source control. Create a new ansible vault to store the AWS Access Key ID and Secret Access Key for a user:

```
(virtualenv)# ansible-vault create [vault_name].yaml
New Vault password:
Confirm New Vault password:
```

```
aws_access_key: AKIAJLHNMCBOITV643UA
aws_secret_key: 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
~
~
~
```

- This will open a text editor. Ensure to utilize `aws_access_key:` and `aws_secret_key:` as variables and provide the respective values for each. Once saved, the file's content will be encrypted.

```
root@ansible:/home/thastings/dnstool# cat test.yml
$ANSIBLE_VAULT;1.1;AES256
6330333862336639636331346335337353263613464653030306236366639666565363438346563
3863303236333437373630353734356636656565343338660a616634633435323237353935393564
35306665663262323463643036373366653765313063326166613334356533613038616430623631
6537376532323534320a633932343734306266343233636336393837666563353033386532636365
6233
```

Run an Ansible Playbook

Run an Ansible Playbook by executing the following:

```
(virtualenv)# ansible-playbook -i hosts --ask-vault-pass dns_server_playbook.yml
```

AWS EC2 Elastic IP Utility

Execute the Elastic IP by running the following:

```
(virtualenv)# allocate.py -i [Instance ID] -t [time]
```