



LAYERS OF CLOUD

Azure and the (Mis-)Storage of Secrets

C:\> WHOAMI

Katie Knowles

- Sr. Manager, Threat Response @ Big 4
(Here with a personal talk, not as my employer)
- Current & previous focus on Cloud + Azure

Previously:

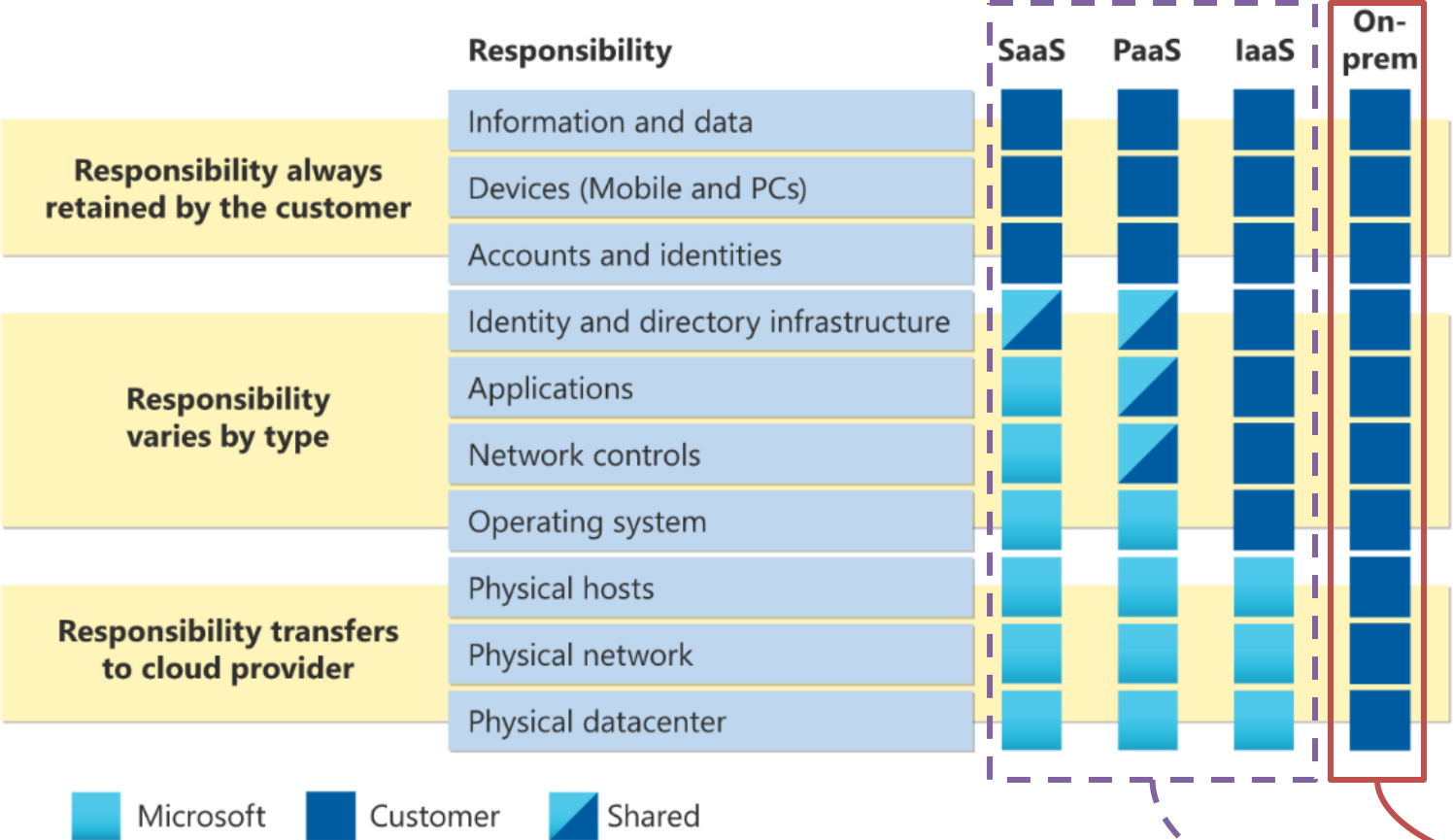
- Pentested Fortune 500s (Cloud + Net + Web)
- Helped run a bug bounty program
- Secured aerospace networks 🚀
- Engineering graduate (BS EE)

Certs:

- OSCP, GPEN, GCSA, Azure Administrator Associate, Azure Security Engineer Associate.

[linkedin.com/in/kaknowles/](https://www.linkedin.com/in/kaknowles/)
twitter.com/_sigil

CLOUDS ON THE HORIZON:



- On-premises, system responsibility is clear.
- Cloud responsibility is mixed by service.
- SaaS, PaaS, IaaS all have different responsibility models.
- Configuring hundreds of services securely across 3 responsibility models... Impossible?

On-Prem = All Business Ownership

Cloud = It Depends??

WHY SPEND TIME ON THIS?

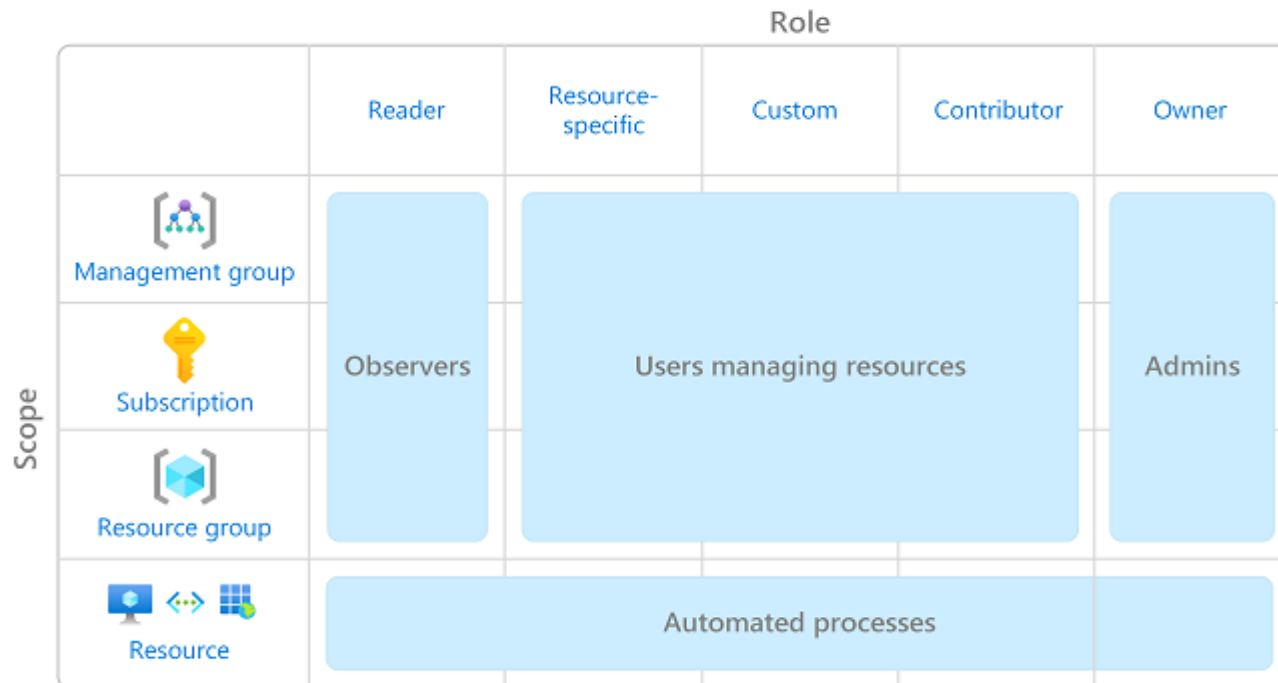


- Let's talk about how Shared Responsibility applies to passwords...
- When attackers are on the network, a key target is improperly stored passwords: Open fileshares, wikis, scripts, etc.
- "Pivoting" with exposed credentials is a key focus of attacks and pentesting.
- What would be the equivalent attack for this type of pivoting in Azure?
- How can we store secrets correctly in Azure, to keep them from this exposure?

AZURE SECRETS

1. AZURE REFRESH (BASICS)
2. STORING SECRETS CORRECTLY & STEALING PASSWORDS
3. STORING SECRETS INCORRECTLY
4. FINDING INSECURE PASSWORDS
5. SECURITY CONCERNS & VULNS

AZURE REFRESH (BASICS)



Role	View	Change	Assign Permissions
Reader	✓	✗	✗
Contributor	✓	✓	✗
Owner	✓	✓	✓

Azure is organized into...

- Management Groups (MGs)
- Subscriptions
- Resource Groups (RGs)
- Resources

Users can have permissions to:

- **Reader:** Access resource details, e.g. settings & status.
- **Contributor:** Change resource settings, access some secrets.
- **Owner:** Assign permissions resources at various levels.

Reader: Seems less risky? (More on that later...)

Contributor / Owner: Most damaging roles for an attacker



SECRETS STORAGE & THEFT

Storing secrets correctly, & what access is still possible.

STORING SECRETS: CORRECT EXAMPLES

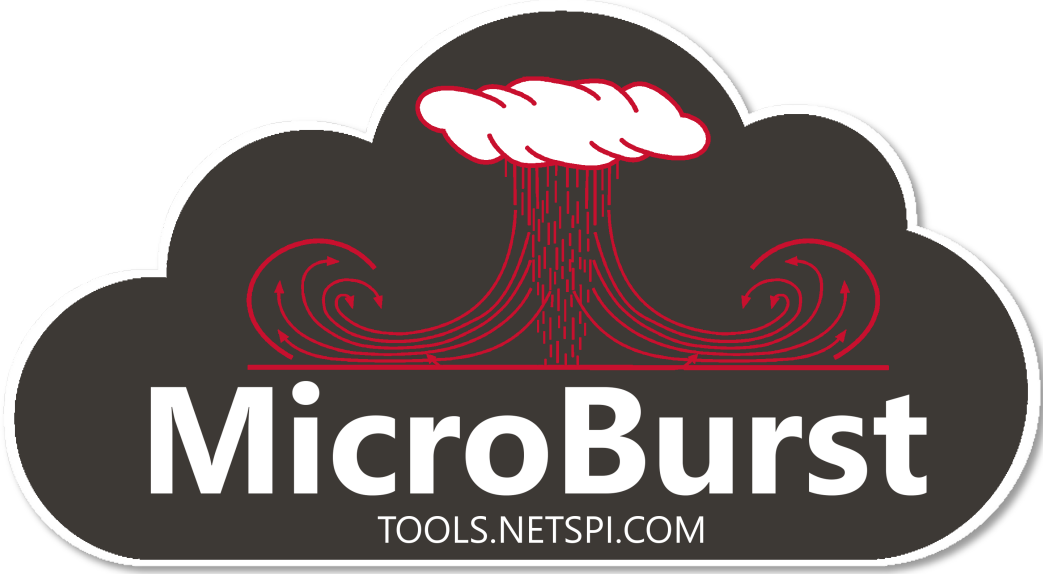
Type of Storage	What Is It?	Who Has Access?
Key Vault	Allows secure storage of passwords, certificates, and access keys. Can be used in automations, or for general password storage.	Users with specific permissions to secrets in the Vault, as well as those with specific permissions to manage the Vault.
Storage Account Keys	Created when Azure Storage is created. Keys allow access to data in the storage account.	Contributors & Owners of the Storage Account.
Service Principal Credentials	Certificates or secrets are stored in protected properties of the Service Principal used for an Azure Application.	Key / Certificate is displayed just once after it is initially set for the Service Principal.
Managed Identity	Some Azure services allow use of a Managed Identity, which performs actions in the background to manage resources.	Users do not have access to Managed Identity credentials.

Some examples (*non-exhaustive*) of how secrets are handled in Azure.

General guidelines include:

- Wherever possible, let the Azure service (Managed Identities, Storage Keys) handle secrets to reduce the risk of exposure.
- When this is not possible, use a service that securely stores secrets (e.g. Key Vault).

STEALING (SECURE) PASSWORDS



Project:

<https://github.com/NetSPI/MicroBurst>

Intro to MicroBurst:

<https://www.netspi.com/blog/technical/cloud-penetration-testing/a-beginners-guide-to-gathering-azure-passwords/>

Attempts recovery of passwords & secrets from:

- Key Vault Passwords & Data
- Storage Account Keys
- Automation Account Credentials
- CosmosDB Account Keys
- App Services Configuration
- Azure Container Registry Admin Passwords
- Azure Kubernetes Service Account Passwords

Overall, these are locations where secrets *should* be stored. They require **Contributor** or **Owner** rights on a resource to be accessed.



INSECURE SECRETS

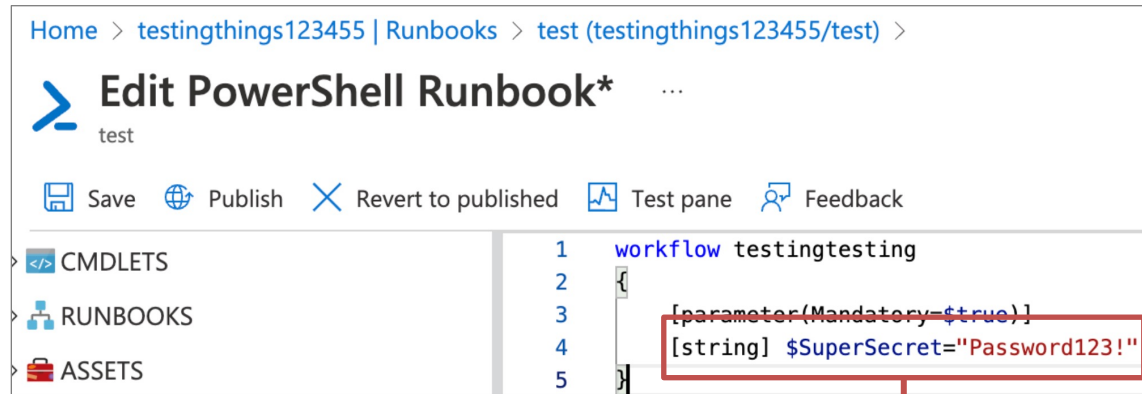
Storing secrets incorrectly, & reviewing Azure resources for them.

STORING SECRETS: INCORRECT EXAMPLES

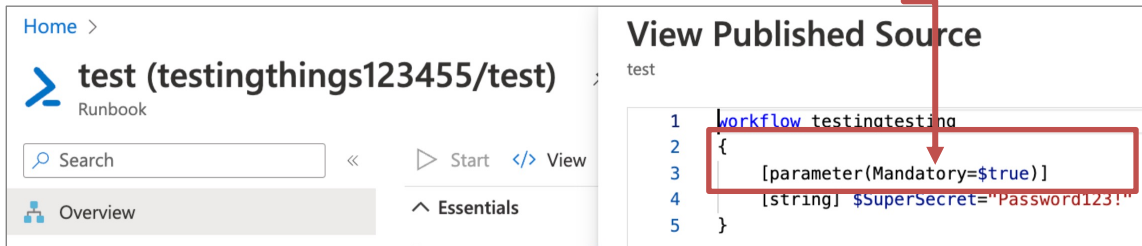


- Users & businesses look to the cloud to build new solutions fast.
- Moving fast can mean misconfigurations, or putting data wherever is quickest.
- Think on-premises hardcoded passwords in scripts... in the cloud.
- The following incorrect examples were identified from a review of Microsoft documentation and developer blogs, and tested in a cloud lab.

RUNBOOK PARAMETERS



```
1 workflow testingtesting
2 {
3     [parameter(Mandatory=$true)]
4     [string] $SuperSecret="Password123!"
5 }
```



```
1 workflow testintesting
2 {
3     [parameter(Mandatory=$true)]
4     [string] $SuperSecret="Password123!"
5 }
```

- Azure Automation Runbooks use PowerShell, Python, or graphical scripts to automate tasks.
- Custom runbooks allow parameters to be set for use as variables at runtime.
- Parameters are exposed in runbook contents (code) published by the creator.
- Users with the **Reader** role can access runbook code, and view any parameters.

TO FIX THIS:

Store secrets in Key Vault or use the “Credentials” section of “Shared Resources” for the Automation Account. Do not store directly in code.

<https://learn.microsoft.com/en-us/azure/automation/shared-resources/credentials?tabs=azure-powershell>

<https://microsoft-bitools.blogspot.com/2019/12/use-azure-key-vault-for-automation.html>

ACI ENVIRONMENT VARIABLES

Create container instance ...

Basics Networking **Advanced** Tags Review + create

Configure additional container properties and variables.

Restart policy On failure

Environment variables

Mark as secure	Key	Value
No	password	P@ssword!
Yes	password2	*****
No		

Resource JSON

testcontainers1234

Resource ID

/subscriptions/

```
1 {
2   "properties": {
3     "sku": "Standard",
4     "provisioningState": "Succeeded",
5     "containers": [
6       {
7         "name": "testcontainers1234",
8         "properties": {
9           "image": "mcr.microsoft.com/azuredocs/aci-helloworld:latest",
10          "ports": [
11            {
12              "protocol": "TCP",
13              "port": 80
14            }
15          ],
16          "environmentVariables": [
17            {
18              "name": "password",
19              "value": "P@ssword!"
20            },
21            {
22              "name": "password2"
23            }
24          ]
25        }
26      }
27    ]
28  }
29 }
```

- ACI provides a feature to pass environment variables into a container at runtime.
- This can make them a popular place to store credentials needed for a container.
- Environment variables are stored in the JSON definition of the resource, under “properties.environmentVariables”.
- Users with the **Reader** role can access Properties not marked as.

TO FIX THIS:

Ensure all sensitive values are marked as “Secure” in the Azure Portal GUI, or use the “secureValue” type.

AZURE LOGIC APPS

The screenshot shows the Azure Logic App Designer interface. At the top, there's a search bar for "Microsoft Azure" and a navigation breadcrumb "exercise-test-app-123 > testflow". The main workspace displays a workflow named "testflow | Designer" with a search bar and "Save" and "Discard" buttons. A "Parameters" button is highlighted with a red box. Below the workflow, there's a "Developer" section with tabs for "Code" and "Designer". The "Designer" tab is active, showing an "HTTP" action. A red box highlights the "Parameters" button, and a red arrow points from it to a "password" parameter definition. The "password" parameter is defined with the following fields:

- Name: password
- Type: String
- Value: P@ssword!

The "Value" field is also highlighted with a red box. Below the parameter definition, there's a "Create parameter" button. The bottom part of the screenshot shows the "Authentication" section of the HTTP action, with a "Password" field containing "P@ssword!". A dashed purple box highlights the "Authentication" section, and a red arrow points from the "Parameters" button to the "Password" field.

- Azure Logic Apps create “low code” automated workflows to integrate apps (e.g. API with a DB).
- “Parameters” as “definition.parameters”.
- “Actions” can include a password field for HTTP auth, as “definition.triggers.HTTP.inputs.authentication.password”.
- Users with the **Reader** role can access Logic App code (visual & generated).

TO FIX THIS:

Use Key Vault parameters as inputs to Logic App actions (functions), and obfuscate sensitive inputs.

<https://365bythijs.be/2021/11/16/interacting-with-key-vault-from-logic-apps-securely/>

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-securing-a-logic-app?tabs=azure-portal#obfuscate>



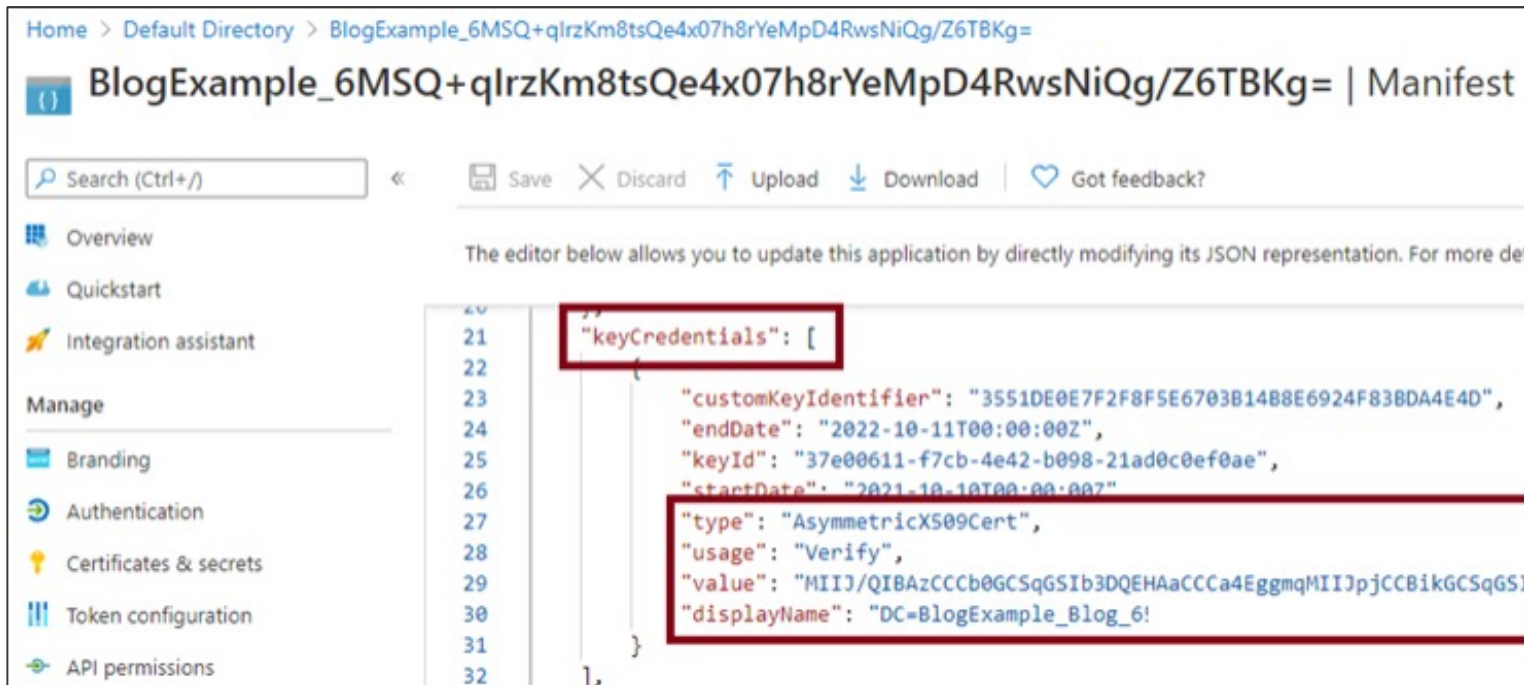
SECURITY CONCERNS

SECURITY CONCERNS



- When we share responsibility, we place trust in our cloud provider. But...
- **Vulns:** Sometimes, a service may experience vulnerabilities that put data at risk in a way that wasn't possible before.
- **Configurations:** Other services may be configured in a way that causes new exposure to support a feature.
- Keeping an eye out for these distinctions is important to protect your cloud (and secrets) at scale.

CERTIFICATE EXPOSURE (CVE-2021-42306)



```
21 "keyCredentials": [  
22   {  
23     "customKeyIdentifier": "3551DE0E7F2F8F5E6703B1488E6924F83BDA4E4D",  
24     "endDate": "2022-10-11T00:00:00Z",  
25     "keyId": "37e00611-f7cb-4e42-b098-21ad0c0ef0ae",  
26     "startDate": "2021-10-10T00:00:00Z",  
27     "type": "AsymmetricX509Cert",  
28     "usage": "Verify",  
29     "value": "MIIJ/QIBAzCCCb0GCSqGSIB3DQEHAaCCCa4EgmgMIIJpjCCBikGCSqGS...",  
30     "displayName": "DC-BlogExample_Blog_6!"  
31   }  
32 ]
```

(Source: NetSPI blog post on vulnerability, pre-remediation)

- Service Principal Accounts provide identities for apps & automation in Azure.
- Base64 private certificates for some accounts were stored in the unprotected “keyCredentials” property of the application manifest, and visible to most AAD users.
- These accounts commonly hold high privileges in Azure subscriptions.

ACI CONTAINER ESCAPE (CVE-2019-5736)

```
provisioningState": "Succeeded",
"resourceGroup": "yuval-test-con-inst",
"restartPolicy": "Always",
"tags": {},
"type": "Microsoft.ContainerInstance/containerGroups",
"volumes": null
}
yavrahami@M-C02YT7FTLVDQ:~$ az container exec --name breakout-ctr --exec-command "/bin/bash"
# ps aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  32.8  4.9 1192968 713280 ?        Rsl   Jul09 2165:07 /hyperkube api
root        20  0.0  0.0    4340    744 ?        Ss    09:08   0:00 /bin/bash
root        25  0.0  0.0   17504   2012 ?        R+    09:08   0:00 ps aux
# hostname
k8s-master-a03wcusl-0
```

(Unit42 video of vulnerability, pre-remediation)

<https://msrc-blog.microsoft.com/2021/09/08/coordinated-disclosure-of-vulnerability-in-azure-container-instances-service/>
<https://unit42.paloaltonetworks.com/azure-container-instances/>
<https://unit42.paloaltonetworks.com/breaking-docker-via-runc-explaining-cve-2019-5736/>

- 2019 CVE, unpatched in 2021.
- Azure Container Instances (ACI) provides managed Kubernetes services, in which containers run on shared multi-tenant environments.
- Unpatched Docker vuln allowed a malicious container to break out of isolation and escalate to cluster admin.
- Allowed code execution on other ACI resources in the same multi-tenant cluster as the malicious container image.

BRUTE-FORCE & ACCOUNT ENUMERATION



Azure AD (AAD) SSO can allow the following attacks from the internet...

- Password brute-force attacks against single-factor accounts
- Username enumeration against multi-factor accounts.

A Smart Lockout policy is required to protect accounts from this activity.

Error Code	Error Description
AADSTS50126	Valid user with an invalid password
AADSTS50056	Valid user that does not have a password in Azure AD
AADSTS50076	Valid credentials but you need MFA to connect to this resource
AADSTS50014	Valid user but the maximum pass-through authentication time has exceeded
AADSTS50034	Not a valid user
AADSTS50053	Account is locked

<https://raxis.com/blog/metasploit-azure-ad-login> <https://arstechnica.com/information-technology/2021/09/new-azure-active-directory-password-brute-forcing-flaw-has-no-fix/>

IN CONCLUSION...

Shared Responsibility is YOUR Responsibility.

Cloud services offer new & exciting business terrain, but have complex settings that may cause users to configure them wrong. Even when responsibility is "shared", the business still has more responsibility than they might realize.

What to do?

- **Educate Users** not to store passwords in insecure areas: scripts, runbooks, and environment variables. Document how to handle secure secrets (e.g. Key Vault, Storage Keys).
- **Remember Shared Responsibility**, and how each cloud service requires different policies to be considered secure.
- **Stay Aware** to new Azure services users adopt, and what shared responsibility models apply to them.

Keep Learning: AZ-900 & AZ-104 Learning Paths are perfect for building your Azure skills, even if you're not taking the exam! 📝

[linkedin.com/in/kaknowles/](https://www.linkedin.com/in/kaknowles/)
twitter.com/_sigil