RSAC | 2025 Conference

Many Voices.
One Community.

SESSION ID: CLS-T02

Canary in the Cloud Mine: Stealthy Tripwires to Detect Post-Breach Activity

Jenko Hwong

Threat Research | Products
WideField Security
https://www.linkedin.com/in/jenkohwong | https://github.com/edleft/rsac2025

Disclaimer

Presentations are intended for educational purposes only and do not replace independent professional judgment. Statements of fact and opinions expressed are those of the presenters individually and, unless expressly stated to the contrary, are not the opinion or position of RSA Conference LLC or any other co-sponsors. RSA Conference LLC does not endorse or approve, and assumes no responsibility for, the content, accuracy or completeness of the information presented.

Attendees should note that sessions may be audio- or video-recorded and may be published in various media, including print, audio and video formats without further notice. The presentation template and any media capture are subject to copyright protection.

© 2025 RSA Conference LLC or its affiliates. The RSAC and RSAC CONFERENCE logos and other trademarks are proprietary. All rights reserved.

Canary in Cloud Mine

Stealthy Tripwires to Detect Post-Breach Activity

Agenda

- Why?
- Deceptive defenses
- Cloud is different
- Examples
- A research and design approach
- Implementation challenges

Blizzard 3/8/24

Attackers are winning with stealthy techniques

Russian threat actors attacking Microsoft using OAuth apps for privilege escalation & persistence

Malicious use of OAuth applications

Threat actors like Midnight Blizzard compromise user accounts to create, mod and grant high permissions to OAuth applications that they can misuse to hide malicious activity. The misuse of OAuth also enables threat actors to maintain access to applications, even if they lose access to the initially compromised account. Midnight Blizzard leveraged their initial access to identify and compromise a legacy test OAuth application that had elevated access to the Microsoft corporate environment. The actor created additional malicious OAut applications. They created a new user account to grant consent in the Microso corporate environment to the actor controlled malicious OAuth applications. T threat actor then used the legacy test OAuth application to grant them the Of 365 Exchange Online *full_access_as_app* role, which allows access to mailboxes

rough password spray

tilized password spray attacks that successfully compr on test tenant account that did not have multifactor enabled. In a password-spray attack, the adversary a me of accounts using a small subset of the most p

https://www.microsoft.com/en-us/security/blog/2024/01/25/midnight-blizzard-guidance-for-responders-on-nation-state-attack/

Storm-0558 7/11/23

Attackers are winning with stolen keys, forged tokens and chained attacks.

Chinese threat actors attacking Microsoft using MSA & enterprise keys for privilege escalation & persistence

Microsoft mitigates China-based threat actor Storm-0558 targeting of customer email

Details

Microsoft investigations determined that Storm-0558 gained access to customer email accounts using Outlook Web Access in Exchange On (OWA) and Outlook.com by forging authentication tokens to access user email.

The actor used an acquired MSA key to forge tokens to access OWA and Outlook.com. MSA (consumer) keys and Azure AD (enterprise) key issued and managed from separate systems and should only be valid for their respective systems. The actor exploited a token validation iss impersonate Azure AD users and gain access to enterprise mail. We have no indications that Azure AD keys or any other MSA keys were use this actor. OWA and Outlook.com are the only services where we have observed the actor using tokens forged with the acquired MSA key.

https://msrc.microsoft.com/blog/2023/07/microsoft-mitigates-china-based-threat-actor-storm-0558-targeting-of-customer-email/

Storm-0558 7/11/23

Attackers are winning with stolen keys, forged tokens and chained attacks. Defense is hard.

Chinese threat actors attacking Microsoft using MSA & enterprise keys for privilege escalation & persistence

1. In April 2021, a system used to generate signing keys for the consumer Microsoft services crashed and generated a Windows crash dump. These dumps should never include cryptographic keys, but in this case, an unspecified bug ("race condition") allowed the key material to be copied from RAM to the crash dump file.
2. The system that's supposed to scan crash dumps for sensitive data didn't detect the presence of the key data in the dump file, again because of an unspecified bug.
3. As part of normal debugging, the crash dump was moved to Microsoft's corporate network that's connected to the Internet. The corporate network is where Microsoft personnel, including engineers, work. The credential-scanning tools on the debugging network didn't detect the key data there either. Microsoft didn't say that this was because of a bug, probably because their credential scanner is looking for actual credentials and not just key material.
4. Microsoft updated a set of internal libraries used for validating cryptographic signatures. These libraries were meant to be shared between their consumer and enterprise services, but because they did a bad job of the updates, the libraries didn't properly enforce the intended separation of scope between the consumer and enterprise services. I'll score this as another bug.
5. Because Microsoft did a bad job of documenting the changes (the ones they did a bad job of implementing), "developers in the mail system" relied on the (non-working) libraries to catch scope violations of the type the attackers used.

Microsoft mi
rgeting of

Details

Microsoft investigation
(OWA) and Outlook.co

The actor used an acqu
issued and managed fr
impersonate Azure AD
this actor. OWA and O

cloud including gov
by using forged au
completed mitiga

ess in Exchange On

AD (enterprise) key
oken validation issu
MSA keys were use
acquired MSA key.

https://practical365.com/storm-0558-snafus/

Storm-2372 2/14/25

Attackers are winning by exploiting OAuth, PRTs, device registration trust, and SSO.

Russian threat actors attacking organizations using OAuth device code phishing, device registration to obtain Primary Refresh Token

Storm-2372 phishing lure and access

Storm-2372's device code phishing campaign has

Observed early activity indicat

using third-party me

Teams, falsely posing

rapport before sendin

phishing emails.

Jan 30., Thu

Hello Mr. [redacted]

My name is [redacted]. I am a researcher at the International Institute for Strategic Studies. From February 24 to 26, the Institute is holding a videoconference on the topic "The future of security in Europe." We invite you to take

February 14, 2025 update:

Within the past 24 hours, Microsoft has observed Storm-2372 shifting to using the specific client ID for Microsoft Authentication Broker in the device code sign-in flow. Using this client ID enables Storm-2372 to receive a refresh token that can be used to request another token for the device registration service, and then register an actor-controlled device within Entra ID. With the same refresh token and the new device identity, Storm-2372 is able to obtain a Primary Refresh Token (PRT) and access an organization's resources. We have observed Storm-2372 using the connected device to collect emails.

transparency, as this is my private number) Thank

https://www.microsoft.com/en-us/security/blog/2025/02/13/storm-2372-conducts-device-code-phishing-campaign/

We don’t know
what we don’t know

But we do know sophisticated techniques continue to evolve…

- Enumeration without logging (N.Frichette DC31 '23, Rhinolabs pacu, Sharepoint/OneDrive enumeration @nyxgeek '20/'23)
- Access/Persistence by exploiting GCP Cloud Shell open ports, cached credentials, persistence (H.Lin DC32 '24, Netskope DC28 '20)
- Phishing/Access/Persistence/Privesc by compromising OAuth flows, SSO (Storm '23, Blizzard '24, @DrAzureAD/Dr. Nestori Synamaa '20, SecureWorks DC30 '22) **Storm-2372 2/14/25**
- Access/Privesc by exploiting Google server-side scripting and loose OAuth permissions (@iamandatory DC29 '21)
- Access/Persistence/Privesc by exploiting service principals, managed identities, admin roles, cross-tenant access (S.Metcalf, Z.Mackie, E.Woodruff, K.Fosaaen DC32 '24)
- Evasion/Persistence by exploiting GCP temporary project delete state and OAuth apps/tokens (GCP Ghost Token, Asterix DC31 '23)
- Phishing for Primary Refresh Tokens and Windows Hello keys (https://dirkjanm.io/phishing-for-microsoft-entra-primary-refresh-tokens, '23) **Storm-2372 2/14/25**

Assume Breach

Insanity is doing the same thing over and over again and expecting different results...

Premise
Fight
Stealth
With
Stealth

Attack Chain and TTPs

IaaS Matrix

Below are the tactics and techniques representing the MITRE ATT&CK® IaaS platform. The techniques below are known to target cloud-based infrastructure-as-a-service (IaaS) platforms. IaaS encompasses cloud-hosted infrastructure, such as virtual machines, object storage, databases, and serverless functionality. The Matrix contains information for the IaaS platform.

View on the ATT&CK® Navigator

Version Permalink

layout: side | show sub-techniques | hide sub-techniques | help

Initial Access (3 techniques)	Execution (4 techniques)	Persistence (6 techniques)	Privilege Escalation (4 techniques)	Defense Evasion (9 techniques)	Credential Access (8 techniques)	Discovery (14 techniques)	Lateral Movement (2 techniques)	Collection (4 techniques)	Exfiltration (2 techniques)	Impact (8 techniques)
Exploit Public-Facing Application	Cloud Administration Command	Account Manipulation (3)	Abuse Elevation Control Mechanism (1)	Abuse Elevation Control Mechanism (1)	Brute Force (3)	Account Discovery (1)	Remote Services (2)	Automated Collection	Exfiltration Over Alternative Protocol	Account Access Removal
Trusted Relationship	Command and Scripting Interpreter (1)	Create Account (1)	Account Manipulation (3)	Exploitation for Defense Evasion	Credentials from Password Stores (1)	Cloud Infrastructure Discovery	Use Alternate Authentication Material (2)	Data from Cloud Storage	Transfer Data to Cloud Account	Data Destruction (1)
Valid Accounts (2)	Serverless Execution	Event Triggered Execution	Event Triggered Execution	Impair Defenses (3)	Forge Web Credentials (2)	Cloud Service Dashboard		Data from Information Repositories		Data Encrypted for Impact
	User Execution (1)	Implant Internal Image	Valid Accounts (2)	Modify Authentication Process (3)	Modify Authentication Process (3)	Cloud Service Discovery		Data Staged (1)		Defacement (1)
		Modify Authentication Process (3)		Modify Cloud Compute Infrastructure (5)	Multi-Factor Authentication Request Generation	Cloud Storage Object Discovery				Endpoint Denial of Service (3)
		Valid Accounts (2)		Modify Cloud Resource Hierarchy	Network Sniffing	Log Enumeration				Inhibit System Recovery
				Unused/Unsupported Cloud Regions	Steal Application Access Token	Network Service Discovery				Network Denial of Service (2)
				Use Alternate Authentication Material (2)	Unsecured Credentials (2)	Network Sniffing				Resource Hijacking (2)
				Valid Accounts (2)		Password Policy Discovery				
						Permission Groups Discovery (1)				
						Software Discovery (1)				

MITRE Engage

Cyber Denial prevents or impairs the adversary's operations.

Cyber Deception reveals and conceals deceptive facts and fictions to mislead and confuse the adversary.

When used together with strategic **Planning & Analysis**, they provide a foundation for **Adversary Engagement**.

© 2022 MITRE
PR_21-01759-21 Last updated: 2/28/2022 Version: 1.0

MITRE Engage Matrix

Prepare	Expose		Affect			Elicit		Understand
Plan	**Collect**	**Detect**	**Prevent**	**Direct**	**Disrupt**	**Reassure**	**Motivate**	**Analyze**
Cyber Threat Intelligence	API Monitoring	Introduced Vulnerabilities	Baseline	Attack Vector Migration	Isolation	Application Diversity	Application Diversity	After-Action Review
Engagement Environment	Network Monitoring	Lures	Hardware Manipulation	Email Manipulation	Lures	Artifact Diversity	Artifact Diversity	Cyber Threat Intelligence
Gating Criteria	Software Manipulation	Malware Detonation	Isolation	Introduced Vulnerabilities	Network Manipulation	Burn-In	Information Manipulation	Threat Model
Operational Objective	System Activity Monitoring	Network Analysis	Network Manipulation	Lures	Software Manipulation	Email Manipulation	Introduced Vulnerabilities	
Persona Creation			Security Controls	Malware Detonation		Information Manipulation	Malware Detonation	
Storyboarding				Network Manipulation		Network Diversity	Network Diversity	
Threat Model				Peripheral Management		Peripheral Management	Personas	
				Security Controls		Pocket Litter		
				Software Manipulation				

MITRE Engage

- Deploy Persona(s) and Deceptive Assets to Engagement Environment
  - Build active directory, file/app server
  - Connect client to environment
  - Litter both server and client(s)
  - Establish connectivity that aligns with storyboard and persona(s) (i.e. teleworker with a VPN)

- Establish Monitoring System
- Deploy Monitoring System
- Monitor Operational Activity

MISSION ESSENTIAL TASK LIST (METL)

The following is intended to serve as an exemplary METL for a generic adversary engagement team. The METL is made up of a series of Mission Essential Tasks (METs). These tasks are the core activities that must be completed during the planning, execution, or analysis phases of an engagement operation. These METs should drive progress towards the operational outcome.

- Establish Gating Criteria
  - Identify exit criteria and appropriate escalation procedures
  - Define operational success
  - Establish acceptable level of risk
  - Define acceptable response time
- Create Engagement Narrative
  - Create persona(s)
  - Determine storyboarding
  - Select pocket litter
  - Define pattern of life for persona(s)
- Establish Monitoring System
  - Build out the collection system
  - Include/establish dashboards for default searches/queries
  - Establish additional systems needed for monitoring
- Build Out Victim Windows System
  - Build computer/system(s) to meet the mission objective/ requirements
  - Use Microsoft Deployment Toolkit for operating system deployment
  - Use tools for software provisioning, configuration management, and application-deployment (e.g. Ansible)
- Build Out Victim Linux System
  - Same as Windows but Linux specific

- Deploy Monitoring System to Engagement Environment
  - Deploy collection system
  - Establish security controls
- Deploy Persona(s) and Deceptive Assets to Engagement Environment
  - Build active directory, file/app server
  - Connect client to environment
  - Litter both server and client(s)
  - Establish connectivity that aligns with storyboard and persona(s) (i.e. teleworker with a VPN)
- Monitor Operational Activity
  - Conduct overwatch and observation
  - Identify basic persistence
  - Identify network activity of interest
  - Identify probable lateral movement
- Forensically Investigate Victim Post Operation
  - Collect disk and memory image
  - Process disk/memory image in to establish timeline of artifacts (e.g. log2timeline/plaso)

- Analyze Data from Live Operation and Forensic Investigation
  - Identify artifacts from live operational data in the post-op at rest data
  - Document findings
  - Inform existing threat model and CTI data
- Conduct Open-Source Intelligence (OSINT) Searches Pre/Post Operation
  - Search for related network activity (ip/domain)
  - Search for related on-system activity (e.g. persistence mechanism)

engage@mitre.org

engage.mitre.org

@ MITRE Engage

MITRE's mission-driven teams are dedicated to solving problems for a safer world. Through our public-private partnerships and federally funded R&D centers, we work across government and

MITRE D3FEND

ATT&CK Lookup

Search D3FEND's 718 Artifacts

D3FEND Lo

Model	– Harden						+ Detect						
+	**Agent Authentication**	**Application Hardening**	**Credential Hardening**	**Message Hardening**	**Platform Hardening**	**Source Code Hardening**	**File Analysis**	**Identifier Analysis**	**Message Analysis**	**Network Traffic Analysis**	**Platform Monitoring**	**Process Analysis**	**User Behavior Analysis**
	Biometric Authentication	Application Configuration Hardening	Certificate Pinning	Message Authentication	Bootloader Authentication	Credential Scrubbing	Dynamic Analysis	Homoglyph Detection	Sender MTA Reputation Analysis	Administrative Network Activity Analysis	File Integrity Monitoring	Database Query String Analysis	Authentication Event Thresholding
	Certificate-based Authentication	Dead Code Elimination	Credential Rotation	Message Encryption	Disk Encryption	Integer Range Validation	Emulated File Analysis	Identifier Activity Analysis	Sender Reputation Analysis	Byte Sequence Emulation	Firmware Behavior Analysis	File Access Pattern Analysis	Authorization Event Thresholding
	Multi-factor Authentication	Exception Handler Pointer Validation	Password Rotation	Transfer Agent Authentication	Driver Load Integrity Checking	Pointer Validation	File Content Analysis	Identifier Reputation Analysis		Certificate Analysis	Firmware Embedded Monitoring Code	Indirect Branch Call Analysis	Credential Compromise Scope Analysis
	Password Authentication	Pointer Authentication	One-time Password		File Encryption	Memory Block Start Validation	File Content Rules	Domain Name Reputation Analysis		Active Certificate Analysis	Firmware Verification	Process Code Segment Verification	Domain Account Monitoring
	Token-based Authentication	Process Segment Execution Prevention	Strong Password Policy		RF Shielding	Null Pointer Checking	File Hashing	File Hash Reputation Analysis		Passive Certificate Analysis	Peripheral Firmware Verification	Process Self-Modification Detection	Job Function Access Pattern Analysis
		Segment Address Offset Randomization			Software Update	Reference Nullification		IP Reputation Analysis		Client-server Payload Profiling	System Firmware Verification	Process Spawn Analysis	Local Account Monitoring
		Stack Frame Canary Validation			System Configuration Permissions	Trusted Library		URL Reputation Analysis		Connection Attempt Analysis	Operating System Monitoring	Process Lineage Analysis	Resource Access Pattern Analysis
					TPM Boot Integrity	Variable Initialization		URL Analysis		DNS Traffic Analysis	Endpoint Health Beacon	Script Execution Analysis	Session Duration Analysis
						Variable Type Validation				File Carving	Input		

MITRE D3FEND

Isolate				Deceive		Evict			Restore	
Access Mediation	**Access Policy Administration**	**Execution Isolation**	**Network Isolation**	**Decoy Environment**	**Decoy Object**	**Credential Eviction**	**Object Eviction**	**Process Eviction**	**Restore Access**	**Restore Object**
Credential Transmission Scoping	Domain Trust Policy	Application-based Process Isolation	Broadcast Domain Isolation	Connected Honeynet	Decoy File	Account Locking	Disk Formatting	Host Shutdown	Reissue Credential	Restore Configuration
IO Port Restriction	Local File Permissions	Executable Allowlisting	DNS Allowlisting	Integrated Honeynet	Decoy Network Resource	Authentication Cache Invalidation	Disk Erasure	Host Reboot	Restore Network Access	Restore Database
Network Access Mediation	User Account Permissions	Executable Denylisting	DNS Denylisting	Standalone Honeynet	Decoy Persona	Credential Revocation	Disk Partitioning	Process Suspension	Restore User Account Access	Restore Disk Image
LAN Access Mediation		Hardware-based Process Isolation	Forward Resolution Domain Denylisting		Decoy Public Release		DNS Cache Eviction	Process Termination	Unlock Account	Restore File
Routing Access Mediation		Kernel-based Process Isolation	Hierarchical Domain Denylisting		Decoy Session Token		Domain Registration Takedown	Session Termination		Restore Email
Network Resource Access Mediation			Homoglyph Denylisting		Decoy User Credential		File Eviction			Restore Software
Remote File Access Mediation			Forward Resolution IP Denylisting				Email Removal			
							Registry Key Deletion			

A Story

```
$ rm -f /usr/bin/ls
$ cat <<EOF > /usr/bin/ls
#!/bin/sh
shutdown now
EOF
$ chmod +x /usr/bin/ls
$ echo *
bin dev etc home opt sbin tmp usr var
$
```

Concepts

chokepoint

```
$
```

tripwire

```
$ ls
Connection to
104.75.99.206 closed
by remote host.
```

landmine | grenade

```
$ rm -f /usr/bin/ls

$ cat<<EOF >/usr/bin/ls
#!/bin/sh
shutdown now
EOF

$ chmod +x /usr/bin/ls
```

don't kill the good guys

```
$ echo *
bin dev etc home opt
sbin tmp usr var
```

common traversal

hidden trigger

response

no FPs

Cloud Attack Chain

Cloud Attack Chain: Discovery

Cloud Attack Chain: Identity Access

Cloud Attack Chain: Resource Access

Focus Areas

1. TTP: Discovery and Persistence

2. Identity Access (honey/canary tokens)
   - Account (Console, Perm Access Key)
   - Role (Temp Session Keys)

3. Resource Access (honey resources)
   - Storage (S3, EBS, EFS, FSx…)
   - Compute
   - EC2/EKS (Compute)
   - CloudShell (Interactive Compute)
   - Lambda/FarGate (Serverless)
   - Images (AMI/ECR)

   - Databases(RDS, Aurora, DynamoDB, ElastiCache, Neptune, Timestream…)

   - Analytics (Athena, Redshift, Glue, EMR, OpenSearch)
   - Data Lake (Lake Formation, Glue Catalog)

Dimensions

1. Environment Location
   - Internal vs External (what is your perimeter?)
   - EC2 (services) vs CloudShell (interactive)
   - Endpoint (CLI/VPN…is inside your perimeter)
   - Images (service config…AMI / ECR / SOAR / IaaC)

2. Attack Chain Stage and Timing
   - Enumeration (pre vs post auth)
   - Initial Access vs Persistence
   - Access vs Use (compute, images)

3. Decoy Type
   - Purloined vs Honey vs Privileged and/or Eunuch

4. Defense Rotation
   - Dynamic vs static (don’t confuse yourself)
   - External vs internal threats (who knows what when)
   - Open sniff test (even if you know my methods…)

5. FPs: Don’t kill the good guys

Discovery: AWS Policy Enumeration

1. List managed policies attached to the user
   - `aws iam list-attached-user-policies --user-name YourUserName`

2. For each managed policy (e.g., AdministratorAccess), get the policy document
   - `aws iam get-policy --policy-arn arn:aws:iam::aws:policy/AdministratorAccess`
   - `aws iam get-policy-version --policy-arn arn:aws:iam::aws:policy/AdministratorAccess --version-id v1`

3. List inline policies attached to the user
   - `aws iam list-user-policies --user-name YourUserName`

4. For each inline policy (e.g., MyInlinePolicy1), get the policy document:
   - `aws iam get-user-policy --user-name YourUserName --policy-name MyInlinePolicy1`

Discovery: AWS Policy Inspection for API Permissions

```
# attached: random_role
{
    "Version": "2012-10-17",
    "Statement": [
        {   "Sid": "230616func",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [

"arn:aws:iam::4973*******90:role/admin_role"
            ]
        },
        {   "Sid": "230615data",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": "arn:aws:s3:::finance.acme.com"
        },
        {   "Sid": "230615dataobj",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": "arn:aws:s3:::test.gleeze.co/*"
        },
        {   "Sid": "roleInspection",
            "Effect": "Allow",
            "Action": [
                "iam:List*",
                "iam:Get*"   ],
            "Resource": [

"arn:aws:iam::4973*******90:role/my_admin_policy",

"arn:aws:iam::4973*******90:role/my_support_policy"
            ]
        },
        {   "Sid": "policyInspection",
            "Effect": "Allow",
            "Action": [
                "iam:List*",
                "iam:Get*"   ],
            "Resource": [

"arn:aws:iam::4973******90:policy/my_admin_policy",

"arn:aws:iam::4973******90:policy/my_support_policy"
            ]
        },
        {   "Sid": "stsCallerId",
            "Effect": "Allow",
            "Action": [
                "sts:GetCallerIdentity"
            ],
            "Resource": "*"
        }
    ]
```

Discovery: AWS Policy Inspection for Trust (Role Trust)

1. List All Roles in the Account
   - `aws iam list-roles --query 'Roles[*].RoleName' --output text`

2. Compare with Role Trust Policies
   - `aws iam get-role --role-name <role-name> --query 'Role.AssumeRolePolicyDocument'`

```
{
  "Effect": "Allow",
  "Principal": {
    "AWS": "arn:aws:iam::123456789012:user/YourUserName"
  },
  "Action": "sts:AssumeRole"
}
```

Chokepoint: AWS Policy

```
# attached: random_role
{
    "Version": "2012-10-17",
    "Statement": [
        {   "Sid": "230616func",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [
                "arn:aws:iam::4973*******90:role/Administrator"
            ]
        },
        {   "Sid": "230615data",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": "arn:aws:s3:::finance.acme.com"
        },
        {   "Sid": "230615dataobj",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": "arn:aws:s3:::finance.acme.com/*"    ]
        },                                                    }
```

tripwire role

tripwire resources

```
{   "Sid": "roleInspection",
    "Effect": "Allow",
    "Action": [
        "iam:List*",
        "iam:Get*"   ],
    "Resource": [
        "arn:aws:iam::4973*******90:role/my_admin_policy",
        "arn:aws:iam::4973*******90:role/my_support_policy"
    ]
},
{   "Sid": "policyInspection",
    "Effect": "Allow",
    "Action": [
        "iam:List*",
        "iam:SetDefaultPolicyVersion",
    "Resource": [
        "arn:aws:iam::4973******90:policy/my_admin_policy",
        "arn:aws:iam::4973******90:policy/my_support_policy"
    ]
},
{   "Sid": "stsCallerId",
    "Effect": "Allow",
    "Action": [
        "sts:GetCallerIdentity"
    ],
    "Resource": "*"
}
```

tripwire perms

Discovery: Azure Activation Role Assignments (PIM/JIT)

```
# List eligible role assignments
GET https://graph.microsoft.com/v1.0/privilegedAccess/microsoftEntra/roleAssignments
```

Chokepoint: Azure Activation Role Assignments (PIM/JIT)


```
{
  "value": [
    {
      "id": "b12345f2-89ab-4cde-9876-234567890abc",
      "roleDefinitionId": "cf123456-7890-4abc-1234-567890abcd12",
      "principalId": "d56789ef-0123-4abc-defg-1234567890gh",
      "principalType": "User",
      "roleDefinitionName": "Cloud App Administrator",
      "assignmentType": "Eligible",
      "status": "Active",
      "startDateTime": "2025-04-01T08:00:00Z",
      "endDateTime": "2025-04-30T23:59:59Z",
      "createdDateTime": "2025-03-30T12:34:56Z",
      "modifiedDateTime": "2025-04-05T09:30:00Z",
      "scope": "/subscriptions/abc123/resourceGroups/AppManagementRG",
      "approvalRequired": true,
      "justificationRequired": false
    },
    {
      "id": "a98765f2-4321-4cde-0987-876543210xyz",
      "roleDefinitionId": "ef123456-7890-4abc-1234-567890abcd12",
      "principalId": "g12345hi-6789-4abc-jklm-0987654321no",
      "principalType": "User",
      "roleDefinitionName": "Privileged Role Administrator",
     …
```


tripwire role

tripwire role

Tripwire: Azure Activation Roles (PIM/JIT)

```
# Activate a role assignment
POST https://graph.microsoft.com/v1.0/privilegedAccess/microsoftEntra/roleAssignments/{id}/activate

# Microsoft Graph API for PIM Events
GET https://graph.microsoft.com/v1.0/privilegedAccess/microsoftEntra/roleAssignments

# PIM Alerts API (Preview)
GET https://graph.microsoft.com/v1.0/privilegedAccess/microsoftEntra/alerts
```

Discovery: Azure Assigned Roles

```
for role in $(az role assignment list --assignee <USER_ID> --query "[].roleDefinitionName" --output tsv); do
    scope=$(az role assignment list --assignee <USER_ID> --query "[?roleDefinitionName=='$role'].scope" --output tsv)
    permissions=$(az role definition list --name "$role" --query "[].permissions" --output json)

    echo "Role: $role"
    echo "Scope: $scope"
    echo "Permissions: $permissions"
    echo "----------------------------------------"
done
```

Role: Contributor

```
Scope: /subscriptions/abc123/resourceGroups/MyResourceGroup
Permissions: [
  {
    "actions": [
      "*"
    ],
    "notActions": [
      "Microsoft.Authorization/*/Delete",
      "Microsoft.Authorization/*/Write"
    ]
  }
]
```

Role: Cloud App Administrator

```
Scope: /subscriptions/abc123/resourceGroups/AppManagementRG
Permissions: [
  {
    "actions": [
      "Microsoft.Entra/applications/*",
      "Microsoft.Entra/servicePrincipals/*",
      "Microsoft.Entra/enterpriseApps/*"
    ],
    "notActions": [
      "Microsoft.Authorization/*"
    ]
  }
]
```

Eunuch Decoy: Azure Deny Assignment (RBAC)

```
# deny-cloud-app-admin.bicep

resource deploymentStack 'Microsoft.Resources/deploymentStacks@2023-06-01' = {
  name: 'RestrictCloudAppAdminStack'
  location: 'eastus'
  properties: {
    denySettings: {
      mode: 'DenyWriteAndDelete'
      excludedActions: [
        "Microsoft.Entra/servicePrincipals/write",
        "Microsoft.Entra/servicePrincipals/delete"
      ]
    }
    actionOnUnmanage: 'detachAll'
  }
}


$ az deployment group create --resource-group MyResourceGroup --template-file deny-cloud-app-admin.bicep
```

Tools: Tripwires

Enumeration

TTP

No Logging

```
$ cat ~/pkgs/pacu/pacu/modules/iam__enum_roles/default-word-list.txt
0
1
2
3
4
5
6
7
8
9
A
ADS
API
APIGateway
AWS
Aaron
Abram
Account
…
Admin
Administrator
Agustin
Ai
Alert_Logic_Cloud_Defender
Alexa
AlexaForBusiness
```

Canary/Honey Tokens: Identity Access

- Access keys (5K users x 2 keys)
- Session tokens (unlimited…)
- Timeouts (EC2 Role 1hr vs IAM User 12hrs)
- EC2 as a canary factor
- Placement (tripwire)

Step-by-step example on an EC2 instance:

1. Determine the EC2 instance ID and other information that might be useful to uniquely identify this server such as application name. You can do this by querying the Instance Metadata Service (IMDS)
2. Use the IAM User credential to AssumeRole into the honeytoken role using a session name of the instance ID and other information as described in 1. Ex. <appname>-i-12313132
3. Receive the temporary credentials back from AWS
4. Write the credentials to a file where an adversary might find them. (~/.aws/credentials)
5. Repeat 2–4 prior to credential expiration (ex. Every 50 minutes)

Netflix Security Team
https://medium.com/@williambengtson/zero-maintenance-aws-canary-tokens-that-scale-b470c6f60da

AWS Compute Credentials: Identity Access

Netflix Security Team
https://medium.com/swlh/active-defense-dynamically-locking-aws-credentials-to-your-environment-47a9c920e704
https://github.com/Netflix-Skunkworks/aws-metadata-proxy

AWS Compute Credentials: Identity Access

Netflix Security Team
https://medium.com/swlh/active-defense-dynamically-locking-aws-credentials-to-your-environment-47a9c920e704
https://github.com/Netflix-Skunkworks/aws-metadata-proxy

AWS Compute Credentials: Identity Access, Neutered

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "*",
            "Resource": "*",
            "Effect": "Deny",
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "52.13.101.11/32"
                    ]
                },
                "StringNotEquals": {
                    "aws:SourceVpc": [
                        "vpc-12345678"
                    ],
                    "aws:SourceVpce": [
                        "vpce-12345678"
                    ]
                }
            }
        },
        {
            "Action": "*",
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::12344567890:role/RoleName",
            "Effect": "Deny"
        }
```

Netflix Security Team
https://medium.com/swlh/active-defense-dynamically-locking-aws-credentials-to-your-environment-47a9c920e704
https://github.com/Netflix-Skunkworks/aws-metadata-proxy

Cached External Credentials: Location

- ~/.aws/credentials
- ~/.azure/msal_token_cache.json

Token Hijacking for API Calls

We just showed how we can easily copy the cached credentials en masse and access the user's GCP environments. We can also pull out the OAuth tokens from the cache and use them directly to execute API calls instead of the CLI.

Let's look back at the sqlite database files in ~/.config/gcloud. The file, access_tokens.db, contains the current OAuth access token, while credentials.db contains the refresh token, the OAuth client id/secret, scopes, and other information.

```
sqlite3 ~/.config/gcloud/access_tokens.db "select access_token from
access_tokens where account_id = 'admin@prod-mfa-hw.com';"
```

```
target-host:~ $ sqlite3 ~/.config/gcloud/access_tokens.db "select access_token from
access_tokens where account_id = 'admin@prod-mfa-hw.com';"
ya29.yt2YtZ2QtMv1JVn7/aO/VUFxt6FrU+ZOviUONXiC9FxtzGZmoEvn3Fdwoisp/+M7LvqCEZeGH+eQOpA
d4cSPZgY9pMX/uf9gzBTYkKPDloBsYUVhQsfv2pZkc0mhlwye+3KIMyaSjh04Z3/zgpdL+boP+u+hZfcJmfV
hYW+WYmYgmw7Y
```

AWS EC2 Resource Access: Timing (Access vs Use)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:StartSession",
        "ssm:DescribeInstanceInformation",
        "ssm:DescribeSessions",
        "ssm:TerminateSession"
      ],
      "Resource": "arn:aws:ec2:region:account-
id:instance/instance-id"
    },
    {
      "Effect": "Allow",
      "Action": "ssm:SendCommand",
      "Resource": "arn:aws:ec2:region:account-
id:instance/instance-id"
    }
  ]
}
```

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "userName": "jdoe",
    "arn": "arn:aws:iam::123456789012:user/jdoe",
    "accountId": "123456789012",
    ...
  },
  "eventTime": "2025-04-12T16:45:31Z",
  "eventSource": "ssm.amazonaws.com",
  "eventName": "StartSession",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "198.51.100.25",
  "userAgent": "ssm-session",
  "errorCode": "AccessDenied",
  "errorMessage": "User: arn:aws:iam::123456789012:user/jdoe is not
authorized to perform: ssm:StartSession on resource: arn:aws:ec2:us-
east-1:123456789012:instance/i-0abcdef1234567890",
  "requestParameters": {
    "target": "i-0abcdef1234567890"
  },
  "responseElements": null,
  "requestID": "b1a1dbe3-d7b1-4e91-917f-aaaaaaaaaaaa",
  "eventID": "c7c1fdd6-5c69-48b7-bc77-bbbbbbbbbbbb",
  "readOnly": false,
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "123456789012"
}
```

AWS S3 Resource Access: id and don’t kill the bad guys

```
aws s3 list-buckets
```

Amazon S3

```
aws s3api list-objects-v2 --bucket your-bucket
```

```
.
└─ digitised/
   └─ b21466154/
      └─ v1/
         ├─ bag-info.txt
         ├─ bagit.txt
         ├─ manifest-sha256.txt
         ├─ ...3 other objects
         └─ data/
            ├─ b21466154.xml
            ├─ b21466154_0001.xml
            ├─ b21466154_0002.xml
            ├─ b21466154_0003.xml
            ├─ alto/
            │  ├─ b21466154_0001_0001.xml
            │  ├─ b21466154_0001_0002.xml
            │  ├─ b21466154_0001_0003.xml
            │  └─ ...1263 other objects
            └─ objects/
               ├─ b21466154_0001_0001.jp2
               ├─ b21466154_0001_0002.jp2
               ├─ b21466154_0001_0003.jp2
               └─ ...1263 other objects
```

AWS AssumeRole: id and don’t kill the bad guys

```
# attached: random_role
{
    "Version": "2012-10-1
    "Statement": [
        {   "Sid": "2306
            "Effect": "Al
            "Action": [
                "sts:Assu
            ],
            "Resource":

"arn:aws:iam::4973******
            ]
        },
        {   "Sid": "2306
            "Effect": "Al
            "Action": [
                "s3:Get*"
                "s3:List
            ],
            "Resource": "
        },
        {   "Sid": "2306
            "Effect": "Al
            "Action": [
                "s3:Get*"
                "s3:List
            ],
            "Resource": "
        },
```

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIamUserAssumeRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::123456789012:user/test
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/Project": "*",
                    "aws:RequestTag/CostCenter": "*",
                    "aws:RequestTag/Department": "*"
                },
                "StringEquals": {"sts:ExternalId": "Example987"}
            }
        },
```

AWS AssumeRole: id and don’t kill the bad guys

```
# attached: random_role
{
    "Version": "2012-10-17",
    "Statement": [
        {   "Sid": "230616func",
            "Effect": "Allow",
            "Action": [
                                    explicit
            ],
            "Re
"arn:aws:iam::4
            ]
        },
        {   "Si
            "Ef
            "Ac
            ],
            "Resource": "arn:aws:s3:::finance.acme.com"
        },
        {   "Sid": "230615dataobj",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": "arn:aws:s3:::test.gleeze.co/*"
        },
```

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role1 \
--role-session-name Session1 \
--tags Key=Star,Value=1 Key=Heart,Value=1 \
--transitive-tag-keys Star Heart
```

```
{   "Sid": "roleInspection",
    "Effect": "Allow",
    "Action": [
                                    _policy",
"arn:aws:iam::4973*******90:role/my_support_policy"
    ]
},
{   "Sid": "policyInspection",
    "Effect": "Allow",
    "Action": [
```

Cloud

chokepoint

- discovery: enumeration actions (List*) /
- resource access
- identity access: compute instances, cloud shells...
- Location: cached external creds

tripwire

- multiple resource tripwires (purloined signposts)
- default admin (privileged)
- fake admin (honey)
- mitm: restrict and monitor VM creds (neutered)
- CLI, cloud shell honey credentials

landmine | grenade

- revoke access
- alerting
- increase logging and monitoring
- risk range with incremental risk adjustments

don't kill the good guys

- multiple signposts: README.txt / MOTD
- interactive, real-time verification
- restricted enumeration
- custom, restricted admin roles
- admin role access (signed/tagged)
- expected changes (CCM, signed/tagged)

Approach

- Design-based (CSP, tools, TTP)
- Enumeration (resources, security principals)
- Access and Use (resources, security principals)
- Common attack vectors (compute environments, external cached credentials, priv esc, data,…persistence)
- Rethink the perimeter (endpoints, internal interactive, images/volumes)
- Integrate across design-dev-devops-admin-ops-hunt-risk (noisy enumeration indicators + downstream activity == higher fidelity, M/L)
- Complementary: Find the good (validated users/actions)

Considerations

- Ignore, Buy, Build
  - Resources / Priorities
  - Motivation / Difficulty
  - Priorities / Money
- Cost and Integration
  - Admin Hooks | Usability
  - CLI | Console | CI/CD
  - Out-of-band management + Audit Logs
  - 3rd-Party Apps (Admin)
- More risk
  - Attack surface and vulns
  - Misconfiguration
  - Malicious insider
  - Rotation
- Coverage and Confidence
  - Research (research, tools, CTFs, breaches, ATT&CK…)
  - Incremental rollout
- Testing and Efficacy
  - Historical Log Analysis
  - Red-team, bug bounties, CTFs
  - Scope: stealth/evasion, persistence, C2

Coverage and Efficacy

IaaS Matrix

Below are the tactics and techniques representing the MITRE ATT&CK® IaaS platform. The techniques below are known to target cloud-based infrastructure-as-a-service (IaaS) platforms. IaaS encompasses cloud-hosted infrastructure, such as virtual machines, object storage, databases, and serverless functionality. The Matrix contains information for the IaaS platform.

View on the ATT&CK® Navigator

Version Permalink

layout: side | show sub-techniques | hide sub-techniques | help

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Exfiltration	Impact
3 techniques	4 techniques	6 techniques	4 techniques	9 techniques	8 techniques	14 techniques	2 techniques	4 techniques	2 techniques	8 techniques
Exploit Public-Facing Application	Cloud Administration Command	Account Manipulation (3)	Abuse Elevation Control Mechanism (1)	Abuse Elevation Control Mechanism (1)	Brute Force (3)	Account Discovery (1)	Remote Services (2)	Automated Collection	Exfiltration Over Alternative Protocol	Account Access Removal
Trusted Relationship	Command and Scripting Interpreter (1)	Create Account (1)	Account Manipulation (3)	Exploitation for Defense Evasion	Credentials from Password Stores (1)	Cloud Infrastructure Discovery	Use Alternate Authentication Material (2)	Data from Cloud Storage	Transfer Data to Cloud Account	Data Destruction (1)
Valid Accounts (2)	Serverless Execution	Event Triggered Execution	Event Triggered Execution	Impair Defenses (3)	Forge Web Credentials (2)	Cloud Service Dashboard		Data from Information Repositories		Data Encrypted for Impact
	User Execution (1)	Implant Internal Image	Valid Accounts (2)	Modify Authentication Process (3)	Modify Authentication Process (3)	Cloud Service Discovery		Data Staged (1)		Defacement (1)
		Modify Authentication Process (3)		Modify Cloud Compute Infrastructure (5)	Multi-Factor Authentication Request Generation	Cloud Storage Object Discovery				Endpoint Denial of Service (3)
		Valid Accounts (2)		Modify Cloud Resource Hierarchy	Network Sniffing	Log Enumeration				Inhibit System Recovery
				Unused/Unsupported Cloud Regions	Steal Application Access Token	Network Service Discovery				Network Denial of Service (2)
				Use Alternate Authentication Material (2)	Unsecured Credentials (2)	Network Sniffing				Resource Hijacking (2)
				Valid Accounts (2)		Password Policy Discovery				
						Permission Groups Discovery (1)				
						Software Discovery (1)				

Testing and Efficacy: Pen-Test (Deception)

2. Engagement Purpose and Objectives

Purpose:

> To evaluate XYZ Corporation's AWS cloud security resilience against advanced threats post-initial access, focusing on **persistence mechanisms**, **privilege escalation opportunities**, and the organization's ability to **detect and respond** to long-term undetected compromises.

Primary Objectives:

Objective	Description
Establish Persistence	Identify and implement stealthy persistence mechanisms within the AWS environment, maintaining access for at least 7 consecutive days without detection.
Privilege Escalation	Attempt to escalate privileges from the initial authenticated user role to obtain administrative access or broader capabilities within AWS accounts.
Stealth Operations	Minimize detection footprint; emulate real-world adversary tactics to evade security monitoring (e.g., GuardDuty, CloudTrail, SIEM solutions).
Risk Validation	Demonstrate impact of successful persistence or escalation on cloud security and business operations.
Recommendations	Provide remediation and detection enhancement recommendations based on observed gaps.

3. Scope Definition

Starting Point:

- The penetration tester will be provided with:
  - AWS IAM **user credentials** with **limited permissions** (normal user role), scoped to pre-defined AWS accounts within XYZ Corporation's cloud environment.

In-Scope Activities:

Category	Actions Permitted
Persistence Techniques	Planting C2 implants (e.g., reverse shells via Lambda, backdoored IAM roles, modified instance userdata)
Privilege Escalation Attempts	Abuse of misconfigured IAM policies, role assumption, misused trust policies, privilege escalation chains
Stealth Techniques	Obfuscation, minimal CloudTrail footprint, log manipulation (within safe guidelines)
Lateral Movement	Cross-account or service-to-service pivoting if possible (within designated AWS accounts only)
Defense Evasion	Testing evasion against security tools (without disabling or breaking them)

Persistence Examples (Allowed):

- Scheduled Lambda executions calling C2 server
- Backdoor IAM users or roles
- SSM Session Manager misuse (if policies permit)
- Implants in EC2 User Data or AMI metadata

Apply What You Have Learned Today

- In the next month
  - Review the slide deck, research, and examples (zero-trust)
  - Evaluate deceptive defense efforts in terms of prioritization, goals, readiness
  - Determine if techniques are applicable to your environment
- In the next 3 months
  - Analyze risk, prioritize gaps
  - Analyze cost / benefit for building own deceptive defenses
  - Prioritize incremental rollouts with focus on low overhead
- In the next 6 months
  - Plan how to test and measure efficacy e.g., integration with red-team exercises
  - Review security vendor approaches to prevention/detection and discuss roadmap on deceptive defensives and signal fidelity

RSAC | 2025 Conference

Many Voices.
One Community.

Jenko Hwong

Threat Research | Products

WideField Security

https://www.linkedin.com/in/jenkohwong

https://github.com/edleft/rsac2025