## Lab 13-01: Azure Reconnaissance with AADInternals

### Scenario

Unisys Corporation, a growing enterprise, has recently migrated its IT infrastructure to Microsoft Azure to enhance scalability and operational efficiency. However, the transition to a cloud-based environment has raised concerns about security risks, particularly unauthorized access and exposure of sensitive data. To strengthen its cloud security posture, UniSys’s IT security team has decided to conduct a reconnaissance assessment of its Azure environment.

### Solution

Unisys Corporation has hired you as a Certified Penetration Tester. As an ethical hacker, your task is to perform Azure reconnaissance using AADInternals. This powerful PowerShell-based tool enables security professionals to extract valuable information from Azure Active Directory (AAD).

AADInternals is primarily intended for auditing and attacking Azure Active Directory (AAD) deployments. However, it may also be used as part of a larger cloud reconnaissance operation. This tool provides many functions, including user enumeration, credential extraction, token extraction and manipulation, privilege escalation, and more.

In this exercise, you will explore techniques to enumerate Azure AD users, groups, and domain details, which attackers could leverage to identify potential entry points. You will use AADInternals to simulate real-world reconnaissance techniques, assess the security posture of Azure AD, and identify misconfigurations that could lead to privilege escalation or unauthorized access.

|  |
| --- |
| 1. Turn on the **Windows 10** virtual machine. Open any browser and go to the following link: **https://drive.google.com/drive/folders/1gdDpPztRPfp-mzipxOF94VXoqao1mQfK** to download AADInternals.  2. In the Windows search, enter **powershell,** and under **PowerShell,** click on **Run as Administrator** to open an administrator PowerShell window. If a **User Account Control** window appears, click **Yes**.    3. In the PowerShell window, execute the following command: **cd C:\Users\Admin\Downloads\AADInternals** to navigate to the AADInternals folder.  4. In the PowerShell window, execute the following command: **Install-Module AADInternals** to install the AADInternals module. In the **Do you want PowerShellGet to install and import the NuGet provider now?** Question type **Y** and press **Enter**. In the Are you sure you want to install the modules from **PSGallery**? question type **A** and press **Enter**.    5. Execute the following command: **Import-Module AADInternals** to import the AADInternals module.    6. Now, we will gather the publicly available information of a target Azure AD, such as Tenant brand, Tenant name, and Tenant ID, along with the names of the verified domains.  7. In the PowerShell window, execute the following command: **Invoke-AADIntReconAsOutsider -DomainName company.com | Format-table**. In this command, replace the company.com with the target company's domain; here, we are using **eccouncil.org**.  8. From the screenshot below, we can gather information such as **DNS**, **MX**, **SPF**, **DMARC**, **DKIM**, etc.    9. Now, we will perform user enumeration in Azure AD. In the PowerShell window, execute the following command: **Invoke-AADIntUserEnumerationAsOutsider -UserName user@company.com.** In this command, replace the **user@company.com** with the target user’s email address.  10. We can see that the result appears **True** under the **Exists** field, which implies that the Azure account with the given username exists, and the attacker can perform further attacks.    11. We can also perform the user enumeration by placing the usernames in a text file by running **Get-Content .\users.txt | Invoke-AADIntUserEnumerationAsOutsider -Method Normal,** where the **users.txt** file contains the target email addresses.  12. Now, to get login information for a domain, execute the following command: **Get-AADIntLoginInformation -Domain company.com**. In this command, replace the company.com with the target company's domain; here, we are using **eccouncil.org**.    13. Now, to get login information for a user, execute the following command: **Get-AADIntLoginInformation -Domain user@company**. In this command, replace the **user@company.com** with the target user’s email address.    14. To get the tenant ID for the given user, domain, or Access Token, execute the following command: **Get-AADIntTenantID -Domain company.com**. In this command, replace the company.com with the target company's domain; here, we are using **eccouncil.org**.    15. To get registered domains from the tenant of the given domain, execute the following command: **Get-AADIntTenantDomains -Domain company.com**. We can see that all the domains associated with the tenant will be listed.    16. Alternatively, you can visit **https://aadinternals.com/osint/** site and enter the tenant ID, domain name, or email to get the openly available information for the given tenant.  17. Open any web browser and go to **https://aadinternals.com/osint/.** Then, enter the domain name in the search box and click on the **Get information** button. Here, we are giving the domain name as **eccouncil.org**.    18. We will get the Domain information and the list of domains connected with the provided domain name.    19. In a similar way, you can enter the tenant ID and email in the search field to view the information regarding the tenant and the user. This concludes the demonstration of Azure reconnaissance with AADInternals. |

## Lab 13-02: Exploit Open S3 Buckets using AWS CLI

### Scenario

As a professional, ethical hacker or penetration tester, you are tasked with assessing the security of S3 buckets in a client’s AWS environment. Misconfigured S3 buckets can leave sensitive data exposed and allow unauthorized actions. These vulnerabilities might enable attackers to manipulate bucket contents, leading to data breaches or malware injections.

In this lab, you will identify publicly accessible S3 buckets, enumerate their contents, and interact with the bucket by creating and deleting a file. This simulation demonstrates how an attacker could exploit insecure configurations to tamper with stored data. The insights gathered will help the client understand the risks and implement robust security measures to protect their cloud assets.

### Solution

As an ethical hacker, you identify S3 buckets with publicly accessible permissions and evaluate the risks associated with them. By enumerating and listing the contents of the bucket, you gain visibility into the types of data exposed. Additionally, by creating a file within the bucket, you simulate an attacker’s ability to insert unauthorized data or inject malicious content. Deleting the file further illustrates the potential for tampering or data loss caused by insecure bucket configurations.

This lab highlights the critical importance of securing S3 buckets by restricting public access, regularly reviewing bucket policies, and enforcing the principle of least privilege to safeguard sensitive information.

|  |
| --- |
| 1. Before starting this task, you must create your AWS account: [**https://aws.amazon.com**](https://aws.amazon.com).   2. Turn on the **ParrotOS** virtual machine and log in with your username and password.  3. Click the **MATE Terminal icon** in the menu to launch the terminal.    4. A ParrotOS **Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the program and enter your password.  **Note**: The password that you type will not be visible.    5. Now, type **cd** and press **Enter** to jump to the root directory.  6. In the terminal window, type **pip3 install awscli --break-system- packages** and press **Enter** to install AWS CLI.    7. Now, we need to configure AWS CLI. To do so, in the terminal window, type **aws configure** and press **Enter**.  8. It will ask for the following details:   * AWS Access Key ID * AWS Secret Access Key * Default region name * Default output format     9. To provide these details, you need to log in to your AWS account.  **Note:** If you do not have an AWS account, create one with the Basic Free Plan and then proceed with the tasks.  10. Click the **AWS account** drop-down menu and click **Security credentials**.    11. Scroll down to the **Access Keys** section.  12. Click the **Create Access Key** button.    13. Now select a use case. We will be selecting the **Command Line Interface (CLI).**    14. Now check the box of confirmation and click on **Next.**    15. Next, click on the **Create access key.**    16. Once the Access key is successfully created, copy the access key and secret access key one by one and paste them into the terminal window. It will prompt you for the default region name; enter **eu-west-1.** The Default output format prompt appears; leave it as the default and press **Enter**.      17. For demonstration purposes, we have created an open S3 bucket with the name **ipscertifiedhacker** in the AWS service. We are going to use that bucket in this lab.  **Note:** The public S3 buckets can be found during the enumeration phase.  18. Now list the directories in the ipscertifiedhacker bucket. In the terminal window, type **aws s3 ls s3://[Bucket Name]** (here, Bucket Name is **ipscertifiedhacker**) and press **Enter.**  **Note:** The bucket name may be different in your lab environment depending on the bucket you are targeting. Do not use this bucket name when you are performing this lab.  19. This will show you the list of directories in the **ipscertifiedhacker** S3 bucket, as shown in the screenshot.    20. Now, maximize the browser window, type **ipscertifiedhacker.s3.amazonaws.com** in the address bar, and press **Enter**.  21. This will show you the complete list of directories and files available in this bucket.    22. Minimize the browser window and switch to **Terminal**.  23. We shall move some files to the ipscertifiedhacker bucket. To do this, in the terminal window, type **echo “You have been hacked” >> Hack.txt** and press **Enter.**  24. By issuing this command, you are creating a file named **Hack.txt**.  25. We will move the **Hack.txt** file to the **ipscertifiedhacker** bucket. In the terminal window, type **aws s3 mv Hack.txt s3://ipscertifiedhacker** and press **Enter.**  26. You have successfully moved the **Hack.txt** file to the **ipscertifiedhacker** bucket.    27. To verify whether the file is moved, switch to the browser window and maximize it. Reload the page.  28. You can observe that the **Hack.txt** file has been moved to the **ipscertifiedhacker** bucket, as shown in the screenshot.    29. Minimize the browser window and switch to the **Terminal** window.  30. Delete the **Hack.txt** file from the **ipscertifiedhacker** bucket. In the terminal window, type **aws s3 rm s3://ipscertifiedhacker/Hack.txt** and press **Enter.**  31. By issuing this command, you have successfully deleted the **Hack.txt** file from the **ipscertifiedhacker** bucket.    32. To verify whether the file is deleted, switch to the browser window and reload the page.  33. The **Hack.txt** file is deleted from the **ipscertifiedhacker** bucket.    34. Thus, you can add or delete files from open S3 buckets.  35. This concludes the demonstration of exploiting public S3 buckets. |

## Lab 13-03: Escalate IAM User Privileges by Exploiting Misconfigured User Policy

### Scenario

As a professional, ethical hacker or penetration tester, you are tasked with assessing the security of a client’s AWS Identity and Access Management (IAM) configurations. Privilege escalation poses a significant threat in the cloud, where even minor misconfigurations in user permission policies can grant users unauthorized access to sensitive resources.

In this lab, you will simulate an attack scenario where you possess an IAM user’s access key and secret access key obtained through social engineering techniques. By analyzing and exploiting the permissions granted to this user, you aim to identify vulnerabilities in the IAM policy. This includes leveraging misconfigured permissions to escalate the user’s privileges to the administrator level, thereby demonstrating the risks associated with overly permissive or improperly designed policies.

### Solution

As an ethical hacker, you exploit misconfigured permissions to escalate privileges in the AWS environment. You begin by creating a custom user policy that provides elevated permissions, such as listing IAM users and S3 buckets. By attaching this policy to the target IAM user account, you demonstrate how improperly configured policies can lead to unauthorized actions.

Next, you verify the escalation by viewing the attached policies of the target user, listing all IAM users in the environment, and enumerating S3 buckets. This process illustrates the security risks associated with misconfigured policies. It highlights the importance of enforcing strict access controls to prevent privilege escalation attacks.

This lab emphasizes the need for continuous policy review, strict access management, and adherence to the principle of least privilege to protect AWS resources from unauthorized access.

**Note:** Before starting this task, create an IAM user (Test) with default settings and create a policy (Test) with permissions including iam:AttachUserPolicy, iam:ListUserPolicies, sts:AssumeRole, and iam:ListRoles, as shown in Figure 19-01. Attackers can exploit these policies to gain administrator-level privileges. After creating the policy, attach the Test policy to the Test user.



Figure 19-01: Test Policy

|  |  |
| --- | --- |
| 1. Before starting this lab, you must create your AWS account: [**https://aws.amazon.com**](https://aws.amazon.com).  2. Turn on the **ParrotOS** virtual machine and log in with your username and password.  3. Click the **MATE Terminal icon** in the menu to launch the terminal.    4. A ParrotOS **Terminal** window appears. In the terminal window, type **sudo su** and press **Enter** to run the program and enter your password.  **Note**: The password that you type will not be visible.    5. Now, type **cd** and press **Enter** to jump to the root directory.  6. In the terminal window, type **pip3 install awscli --break-system-packages** and press **Enter** to install AWS CLI.    7. Now, we need to configure AWS CLI. To do so, in the terminal window, type **aws configure** and press **Enter**.  8. It will ask for the following details:   * AWS Access Key ID * AWS Secret Access Key * Default region name * Default output format     9. To provide these details, you need to log in to your AWS account.  **Note:** If you do not have an AWS account, create one with the Basic Free Plan and then proceed with the lab.  10. Click the **AWS account** drop-down menu and click **Security credentials**.    11. Scroll down to the **Access Keys** section.  12. Click the **Create Access Key** button.    13. Now select a use case. We will be selecting the **Command Line Interface (CLI).**    14. Now check the box of confirmation and click on **Next.**    15. Next, click on the **Create access key.**    16. Once the Access key is successfully created, copy the access key and secret access key one by one and paste them into the terminal window. It will prompt you for the default region name; enter **eu-west-1.** The Default output format prompt appears; leave it as default and press **Enter**.      17. After configuring the AWS CLI, we create a user policy and attach it to the target IAM user account to escalate the privileges.  18. In the terminal window, type **vim user-policy.json** and press **Enter.**  **Note:** This command will create a file named **user-policy** in the root directory.    19. A command-line text editor appears; press **I** to insert and type the script given below:   |  | | --- | | **{**  **"Version": "2012-10-17",**  **"Statement": [**  **{**  **"Effect": "Allow",**  **"Action": "\*",**  **"Resource": "\*"**  **}**  **]**  **}** |   **Note:** This is an AdministratorAccess policy that gives the administrator access to the target IAM user. Ignore the **$** symbols in the script.    20. After entering the script given in the previous step, press the **Esc** button. Then, type **:wq!** and press **Enter** to save the text document.    21. Now, we will attach the created policy (user-policy) to the target IAM user’s account. To do so, type **aws iam create-policy --policy-name user-policy --policy-document file://user-policy.json** and press **Enter**.**Note:** If you receive an error that the policy already exists, rename the file and try again.    22. The created user policy is displayed, showing various details such as **PolicyName**, **PolicyId**, and **Arn**.    23. In the terminal, type **aws iam attach-user-policy --user-name [Target Username] --policy-arn arn:aws:iam::[Account ID]:policy/user-policy** and press **Enter.**    24. The above command will attach the policy (user-policy) to the target IAM user account (here, test).  25. Now, type **aws iam list-attached-user-policies --user-name [Target Username]** and press **Enter** to view the attached policies of the target user (here, test).    26. The result appears, displaying the attached policy name (user-policy), as shown in the screenshot.    27. Now that you have successfully escalated the privileges of the target IAM user account, you can list all the IAM users in the AWS environment. To do so, type **aws iam list-users** and press **Enter.**    28. The result appears, displaying the list of IAM users, as shown in the screenshot.    29. You can also list all the S3 buckets by typing **aws s3api list-buckets --query "Buckets[].Name"**    30. Similarly, you can use various commands to obtain complete information about the AWS environment, such as the list of user policies, role policies, and group policies, as well as to create a new user.   * **User Policies:** aws iam list-user-policies * **Role Policies:** aws iam list-role-policies * **Group policies:** aws iam list-group-policies * **Create user:** aws iam create-user   31. This concludes the demonstration of escalating IAM user privileges by exploiting a misconfigured user policy. |

**Lab 13-04: Vulnerability Assessment on Docker Images using Trivy**

**Scenario**

SoftTech Corporation, a company increasingly adopting containerized applications to streamline development and deployment, has implemented Docker as part of its infrastructure to achieve faster, more efficient software distribution. However, with the rise of containerization, the security of Docker images becomes critical to prevent vulnerabilities from being exploited in production environments. As part of their ongoing security improvement initiative, SoftTrech’s security team has tasked you, as an ethical hacker, with performing a vulnerability assessment on Docker images to identify potential security weaknesses.

**Solution**

SoftTech Corporation hired you as a Certified Penetration Tester. To find vulnerabilities in the Docker images, you will utilize Trivy to analyze and assess the security of Docker containers. You will perform active scanning to detect known vulnerabilities in base images, misconfigurations, and missing patches. By simulating real-world attack techniques, you will understand how these vulnerabilities could be exploited and identify critical weaknesses that could lead to breaches.

Trivy is an advanced security scanner that discovers vulnerabilities and misconfigurations in a variety of environments, including container images, file systems, Git repositories, virtual machine images, Kubernetes, and AWS. Trivy’s extensive scanners detect OS package vulnerabilities, sensitive information, IaC issues, and more, delivering a strong security solution for your infrastructure.

|  |  |
| --- | --- |
| 1. Turn on the **ParrotOS** virtual machine. Open the Terminal and execute the **sudo su** command to run programs with root privileges.    2. Execute the below-provided command to install Trivy.   |  | | --- | | sudo apt-get install wget gnupg  wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null  echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb generic main" | sudo tee -a /etc/apt/sources.list.d/trivy.list  sudo apt-get update  sudo apt-get install trivy |     3. Execute the following command: **docker pull ubuntu** to install the first docker image.    4. Once the image is pulled, we will perform a vulnerability assessment. Execute the following command: **trivy image ubuntu**. We can observe that we have a total of 0 vulnerabilities, and it is completely secure.    5. Now, we will analyze the vulnerable image. Execute the following command: **docker pull nginx:1.19.6** to pull the vulnerable image.    6. Then execute the following command: **trivy image nginx:1.19.6** to scan the image.    7. In the screenshot below, we can see that we have a total of 401 vulnerabilities, which are categorized as well, along with the CVEs mentioned. This concludes the demonstration of vulnerability assessment on docker images using Trivy. |