Vishal@123

71535

Vishu123

vishal.mohindru

Homaxivox123$

vishal.mohindru

Homaxivox123$

vishal.mohindru

oN3Ol!APx762

Homaxivox123$

Homaxivox@123$

Password

Credential

Passwords

Any combination of up to 20,000 characters consisting of:

* a-z (case insensitive)
* 0-9
* forward slashes (/) or plus signs (+)
* Up to two equal signs (=)

for example:

MIIKcQIBAzCCCi0GCSqGSIb3DQEHAaCCCh4EggoaMIIKFjCCBg8GCSqGSIb3DQEHAaCCBgAEggX8MIIF+DCCBfQGCyqGSIb3DQEM

or

Various command line sign-in credentials formats, for example:

-u username:\*\*\*\*\*\*\*\*

or

-u username -p \*\*\*\*\*\*\*\*

or

/f ... /p \*\*\*\*\*\*\*\*

or

-Password \*\*\*\*\*\*\*\*

or

-U username%\*\*\*\*\*\*\*\*

or

-secrets:\*\*\*\*\*\*\*\*

for example:

zDbg.DataPuller.exe -secrets:eyJ

or

Various password formats in code snippets, for example:

new X509Certificates2(

or

ConvertTo-SecureString -String \*\*\*\*\*\*\*\*

or

password = "\*\*\*\*\*\*\*\*"

or

"password" : "\*\*\*\*\*\*\*\*"

or

UserPasswordCredential(

for example:

password = "ZYXWVU\_1";

or

Various password formats in script, for example:

password = \*\*\*\*\*\*\*\*

for example:

password=ZYXWVU\_1

or

Various password formats in XML, for example:

XMLCopy

<secret>\*\*\*\*\*\*\*\*</secret>

<password>\*\*\*\*\*\*\*\*</password>

<setting name="password" value="\*\*\*\*\*\*\*\*" >

<setting name="password">\*\*\*\*\*\*\*\*</setting>

<setting name="password"><value>\*\*\*\*\*\*\*\*</value></setting>

for example:

<secret>ZYXWVU\_1</secret>

or

Any combination of 22 characters consisting of:

* a-z (case insensitive)
* digits, forward slashes, or plus signs
* ends with two equal signs (=)

for example:

abcdefgh0123456789/+AB==

or

Any combination of 32 characters consisting of:

* a-f or A-F (case-sensitive) or 0-9

for example:

abcdef0123456789abcdef0123456789

or

Any combination of 32 characters consisting of:

* a-z (case insensitive)
* 0-9
* forward slashes (/) or plus signs (+)

for example:

abcdefghijklmnopqr0123456789/+AB

or

Any combination of 43 characters consisting of:

* a-z (case insensitive)
* 0-9
* forward slashes (/) or plus signs (+)
* ends with an equal sign (=)

for example:

abcdefghijklmnopqrstuvwxyz0123456789/+ABCDE=

or

Any combination of 86 characters consisting of:

* a-z (case insensitive)
* 0-9
* forward slashes (/) or plus signs (+)
* ends with two equal signs (=)

for example:

abcdefghijklmnopqrstuvwxyz0123456789/+ABCDEabcdefghijklmnopqrstuvwxyz0123456789/+ABCDE==

**Credential example**

| **Confidence Band** | **Example** |
| --- | --- |
| High | password = D3m0P@sswd! |
| Medium | secret : DemoPasswd! |
| Low | password = demopasswd2 |

**Checksum**

Yes

SITs that have checksums use a unique calculation to check if the information is valid. This means when the **Checksum** value is **Yes**, the service can make a positive detection based on the sensitive data alone. When the **Checksum** value is **No** additional (secondary) elements must also be detected for the service to make a positive detection.

**Description**

This SIT is designed to match the security information that's like usernames and passwords used in general sign-in process [user login process](https://learn.microsoft.com/en-us/azure/key-vault/quick-create-portal). It uses several primary resources:

* Patterns of Base64 encoded string literal.
* Patterns of Password context in command line.
* Patterns of Password context in code.
* Patterns of Password context in script.
* Patterns of Password context in XML.
* Patterns of Base64 encoded 128-bits symmetric key.
* Patterns of Hex encoded 128-bits symmetric key.
* Patterns of Base64 encoded 192-bits symmetric key.
* Patterns of Base64 encoded 256-bits symmetric key.
* Patterns of Base64 encoded 512-bits symmetric key.
* Patterns of CredentialName, CredentialFeatures, AccountIdentityName, AccountIdentityValue, ResourceType, ResourceName, ID, AccountName.
* Patterns of mockup values, redactions, and placeholders.
* A dictionary of vocabulary words.

The patterns are designed to match actual credentials with reasonable confidence. The patterns don't match credentials formatted as examples. Mockup values, redacted values, and placeholders, like credential type or usage descriptions, in the position where an actual secret value should present won't be matched.

**Keywords**

**Keyword\_Base64EncodedStringLiteral**

* MII

**Keyword\_PasswordContextInCmdLine**

* certutil
* zdbg
* secret
* VSTS\_TOKEN
* curl
* PowerShell
* ps1
* -u
* Smc
* AutoLogon
* ldifde
* Rclone
* --env
* SignTool
* winexe
* net

**Keyword\_PasswordContextInCode**

* key
* x509c
* credential
* password
* pw
* securestring

**Keyword\_Pas**