NEO APAC HACKATHON

RESILIENCE, MEET INNOVATION

RED4SEC

Scopes & Permissions

Speaker: **Fernando Díaz Toledano**

RED4SEC

Introduction

Security is our primary.

RED4SEC

neo

neo

Summary

RED4SEC

Copyright 2023

RED4SEC

WitnessScope

Each of the signatures of a transaction can have a different scope defined.

Scopes signatures in N3 are important because they allow users to control the scope of their signature at a finer level of granularity.

This means that users can specify **which contracts are allowed to use their signature**, preventing unauthorized contracts from using it.

This helps to reduce the attack surface of a transaction and increases security.

Additionally, users can also specify **WitnessRules**, which allows them to further customize the scope of their signature.

```
[Flags]
public enum WitnessScope : byte
{
    /// <summary>
    /// Indicates that no contract was witnessed. Only sign the transaction.
    /// </summary>
    None = 0,

    /// <summary>
    /// Indicates that the calling contract must be the entry contract.
    /// The witness/permission/signature given on first invocation will automatically expire if
    /// entering deeper internal invokes.
    /// This can be the default safe choice for native NEO/GAS (previously used on Neo 2 as "attach“
    /// mode).
    /// </summary>
    CalledByEntry = 0x01,

    /// <summary>
    /// Custom hash for contract-specific.
    /// </summary>
    CustomContracts = 0x10,

    /// <summary>
    /// Custom pubkey for group members.
    /// </summary>
    CustomGroups = 0x20,

    /// <summary>
    /// Indicates that the current context must satisfy the specified rules.
    /// </summary>
    WitnessRules = 0x40,

    /// <summary>
    /// This allows the witness in all contexts (default Neo2 behavior).
    /// </summary>
    /// <remarks>Note: It cannot be combined with other flags.</remarks>
    Global = 0x80
}
```

Links

https://github.com/neo-project/neo/blob/master/src/Neo/Network/P2P/Payloads/WitnessScope.cs
https://neospcc.medium.com/thou-shalt-check-their-witnesses-485d2bf8375d

WitnessRules

Indicates that the current context must satisfy the specified rules.

The **WitnessRules** is a more complex scope that allows to establish more specific conditions for the permission of the signature in the contract.

WitnessRules are used in N3 to allow users to control the signature scope at a finer level of granularity. This allows users to specify rules and scopes for their signatures, so that the signature can be used only for verifying the specified call, preventing unauthorized contracts from using the user signature.

For example, a user can specify that the signature is allowed only when the contract is *0xef4073a0f2b305a38ec4050e4d3d28bc40ea63f5* and is invoked at entry, by using the logical conjunction of the two expressions.

Links

https://github.com/neo-project/neo/tree/master/src/Neo/Network/P2P/Payloads/Conditions

AndCondition

BooleanCondition

CalledByContractCondition

CalledByEntryCondition

CalledByGroupCondition

GroupCondition

NotCondition

OrCondition

ScriptHashCondition

$$\begin{tikzcd}
\text{WitnessCondition} \arrow[d] \\
\text{WitnessConditionType}
\end{tikzcd}$$

RED4SEC

Read only in Storage

Gets the read only storage context for the current contract.

Use read-only **StorageContext** is crucial due to its fundamental role in data security and integrity.

By using a **read-only StorageContext**, contract information can never be modified, which makes the method that uses it **more resilient** to errors and vulnerabilities.

```
System.Storage.GetReadOnlyContext

Gets the readonly storage context for the current contract.


System.Storage.AsReadOnly

Converts the specified storage context to a new readonly storage context.
```

Links

https://github.com/neo-project/neo/blob/master/src/Neo/SmartContract/StorageContext.cs

CallFlags

Represents the operations allowed when a contract is called.

The **CallFlags** in N3 are important because they define the special behaviors allowed when invoking smart contracts.

The flags allow developers to control the **permissions** of the contract, such as allowing the contract to **modify states**, do **contract calls**, and send **notifications**.

This helps to **ensure** that the contract is **secure**, and that the data is not tampered with.

Links

https://github.com/neo-project/neo/blob/master/src/Neo/SmartContract/CallFlags.cs

```csharp
[Flags]
public enum CallFlags : byte
{
    /// <summary>
    /// No flag is set.
    /// </summary>
    None = 0,

    /// <summary>
    /// Indicates that the called contract is allowed to read states.
    /// </summary>
    ReadStates = 0b00000001,

    /// <summary>
    /// Indicates that the called contract is allowed to write states.
    /// </summary>
    WriteStates = 0b00000010,

    /// <summary>
    /// Indicates that the called contract is allowed to call another contract.
    /// </summary>
    AllowCall = 0b00000100,

    /// <summary>
    /// Indicates that the called contract is allowed to send notifications.
    /// </summary>
    AllowNotify = 0b00001000,

    /// <summary>
    /// Indicates that the called contract is allowed to read or write states.
    /// </summary>
    States = ReadStates | WriteStates,

    /// <summary>
    /// Indicates that the called contract is allowed to read states or call another contract.
    /// </summary>
    ReadOnly = ReadStates | AllowCall,

    /// <summary>
    /// All flags are set.
    /// </summary>
    All = States | AllowCall | AllowNotify
}
```

ContractManifest

When a smart contract is deployed, it must explicitly declare the features and permissions it will use.
When it is running, it will be limited by its declared list of features and permissions and cannot make any behavior beyond the scope of the list.

Three fields concerning contract invocation permission are specified in the contract manifest file: **Groups**, **Permissions**, and **Trust**.

The **wallet determines whether to issue a security warning** to the user based on the settings in the **Groups** and **Trust** fields.

Permissions dictate whether contracts can call each other.

Links

https://github.com/neo-project/neo/blob/master/src/Neo/SmartContract/Manifest/ContractManifest.cs
https://docs.neo.org/docs/en-us/develop/deploy/invoke.html#invocation-permission

```
{
  "name": "NameService",
  "groups": [],
  "features": {},
  "supportedstandards": [ "NEP-11" ],
  "abi": { ... },
  "permissions": [
    {
      "contract": "0x726cb6e0cd8628a1350a611384688911ab75f51b",
      "methods": [ "ripemd160" ]
    },
    {
      "contract": "0xacce6fd80d44e1796aa0c2c625e9e4e0ce39efc0",
      "methods": [
        "atoi",
        "deserialize",
        "serialize",
        "stringSplit"
      ]
    },
    {
      "contract": "0xef4073a0f2b305a38ec4050e4d3d28bc40ea63f5",
      "methods": [ "getCommittee" ]
    },
    {
      "contract": "0xfffdc93764dbaddd97c48f252a53ea4643faa3fd",
      "methods": [ "getContract", "update" ]
    },
    {
      "contract": "*",
      "methods": [ "onNEP11Payment" ]
    }
  ],
  "trusts": [],
  "extra": { ... }
}
```

RED4SEC

Permissions

The permissions of the contract.

ContractPermission allows developers to restrict which contracts their contract can call. This helps to ensure that contracts are secure and that malicious contracts cannot be called.

By declaring Permission first and then invoking, developers can specify which contracts they trust and which methods they can access.

For example, developers can specify that a contract can only access the methods *getBalance* and transfer of the contract with hash *X*, and the method *commonMethodName* of any contract in the group with public key *Y*. This helps to protect the contract's data from being accessed by unauthorized contracts.

Links

https://github.com/neo-project/neo/blob/master/src/Neo/SmartContract/Manifest/ContractPermission.cs

ContractPermission

This field is an array containing a permission object, which defines other contracts and methods that the contract wants to call. The contract can be ScriptHash, Group, or wildcard *. The method is the method name or wildcard *. Contracts or methods not declared in the manifest cannot be called by the contract.

Trusts

The trusted contracts and groups of the contract.
If a contract is trusted, the user interface will not give any warnings when called by the contract.

ContractPermissionDescriptor ensures that contracts can only call other contracts and methods that have been explicitly allowed. This helps to **improve** the **security** of the smart contract and prevents malicious actors from exploiting the contract.

Additionally, it allows developers to create a **trust system** between contracts, which can be used to give security warnings to wallets.

The **ContractPermissionDescriptor** is used to define the constants and methods that are available to the contract. This allows developers to control who can access the contract and what methods can be called, ensuring that only authorized users can interact with the contract.

Links

https://github.com/neo-project/neo/blob/master/src/Neo/SmartContract/Manifest/ContractPermissionDescriptor.cs

ContractPermissionDescriptor

Defines other contracts trusted by the contract. The contract can be ScriptHash, Group, or wildcard *. If a contract is trusted, the user will not receive any warning message when the contract is called.

RED4SEC

THANK YOU FOR YOUR ATTENTION