Cyberscope

Audit Report

SMARTSWAP

November 2023

Network	BTC
Type	Script
Audited by	© cyberscope

Table of Contents

Review

BTC SCRIPT	Files

Audit Updates

Initial Audit	16 Nov 2023

Source Files

Filename	Source
script	local files

Overview

The SMARTSWAP script involves the implementation of a Bitcoin transaction script, tailored to facilitate conditional transactions, potentially within an escrow or multi-party agreement context. The script is a testament to the versatility and power of Bitcoin's scripting language, enabling transactional logic far beyond simple fund transfers. The core of the project revolves around creating a secure, automated system for managing transactions where multiple conditions and parties are involved.

The script utilizes conditional operations (`OP_IF` , `OP_ELSE`), cryptographic hash functions (`OP_HASH160`), and verification checks (`OP_EQUALVERIFY` , `OP_CHECKSIG`) to ensure that specific criteria are met before a transaction is processed. This setup is ideal for escrow services, where funds need to be securely held and released only upon the fulfillment of agreed conditions by the involved parties (such as a buyer and a seller). In such a setup, each party may have unique secrets (conditions) that need to be provided and verified for the transaction to proceed. This ensures that the transaction is executed exactly as intended, with all parties adhering to the pre-agreed terms.

Overall, this project exemplifies how Bitcoin's scripting capabilities can be harnessed to create secure, conditional, and automated transaction processes, suitable for a variety of financial agreements and scenarios. It highlights the potential of blockchain technology in revolutionizing how we approach and execute contractual and financial transactions in a decentralized, secure, and automated manner.

Script Review

The script, detailed below, leverages Bitcoin's scripting language to enable complex and condition-dependent transactions, ensuring that funds are transferred only when specific pre-agreed criteria are met by the involved parties.

```
OP_DUP
OP_1 OP_EQUAL
OP_IF
OP_DROP
    ${buyerPubKeyHash} ${buyerSecretHash}
OP_ELSE OP_2 OP_EQUALVERIFY
    ${sellerPubKeyHash} ${sellerSecretHash}
OP_ENDIF
OP_ROT
OP_HASH160
OP_EQUALVERIFY
OP_OVER
OP_HASH160
OP_EQUALVERIFY
OP_CHECKSIG
```

Script Breakdown

Each operation of the script plays a critical role in ensuring that the transaction conditions are precisely met. The script is meticulously crafted to handle conditional logic, providing a secure and efficient way to manage transactions between parties. Below is a detailed breakdown of each operation in the script and its specific function within the overall transaction process:

1. OP_DUP: Duplicates the top item on the stack.
2. OP_1 OP_EQUAL: Pushes the value 1 onto the stack, then checks if the top two items on the stack are equal.
    - If they are equal, it pushes 1 (true) onto the stack;
    - otherwise, it pushes 0 (false).
3. OP_IF: Conditional operator that executes the following statements only if the top item on the stack is true (non-zero).
4. OP_DROP: Removes the top item from the stack. This is executed only if the condition in OP_IF is true.
5. ${buyerPubKeyHash} ${buyerSecretHash}: Pushes the buyer's public key hash and secret hash onto the stack.
    - This is executed only if the condition in OP_IF is true.
6. OP_ELSE OP_2 OP_EQUALVERIFY: If the condition in OP_IF is false, this part is executed.
    - OP_ELSE marks the start of the else block.
    - OP_2 pushes the value 2 onto the stack.
    - OP_EQUALVERIFY checks if the top two items on the stack are equal and removes them from the stack;
    - if they are not equal, script execution is terminated.
7. {sellerPubKeyHash} ${sellerSecretHash}: Pushes the seller's public key hash and secret hash onto the stack.
    - This is executed only if the condition in OP_ELSE is true.
8. OP_ENDIF: Marks the end of the if-else block.
9. OP_ROT: Rotates the top three items on the stack.
10. OP_HASH160: Computes the RIPEMD160(SHA256()) hash of the top item on the stack.

11. OP_EQUALVERIFY: Checks if the top two items on the stack are equal and removes them from the stack; if they are not equal, script execution is terminated.
12. OP_OVER: Copies the second-to-top item on the stack to the top of the stack.
13. OP_HASH160: Again, computes the RIPEMD160(SHA256()) hash of the top item on the stack.
14. OP_EQUALVERIFY: Again, checks if the top two items on the stack are equal and removes them from the stack; if they are not equal, script execution is terminated.
15. OP_CHECKSIG: Checks the signature of the transaction against the public key on the stack. If the signature is valid, 1 (true) is pushed onto the stack; otherwise, 0 (false) is pushed.

Script Steps

The script is a set of instructions for a Bitcoin transaction, primarily used in scenarios requiring conditional logic, like escrow agreements.

1. Initialization and Verification:

- OP_DUP: The script starts by duplicating the top item on the stack. This item is typically a public key or a condition indicator.
- OP_1 OP_EQUAL: The script then pushes the value 1 onto the stack and checks if this value matches the previously duplicated item. This comparison results in either true (1) if they match or false (0) if they don't.

2. Conditional Logic:

- OP_IF: This conditional operator acts based on the true/false value obtained from the previous step. If true, the script proceeds with the following operations.
- OP_DROP: It then drops the top item from the stack (the true/false indicator), clearing the way for the next set of operations.

3. Buyer's Path (If Condition is True):

- If the condition is true, indicating a specific scenario (like a buyer’s action in an escrow transaction), the script pushes the buyer's public key hash and a secret hash onto the stack.
- This path is designed to verify the buyer’s credentials and secret information.

4. Seller's Path (If Condition is False):

- OP_ELSE OP_2 OP_EQUALVERIFY: If the initial condition is false, the script follows the 'else' path, intended for the seller. It pushes the value 2 onto the stack and verifies if it matches the top stack item.
- Seller's Details: If verification is successful, the script then pushes the seller's public key hash and secret hash onto the stack, similar to the buyer's path.

5. Final Verification and Execution:

- OP_ENDIF marks the end of the conditional logic.

- OP_ROT: The script rotates the top three items on the stack, reordering them for the final verification steps. Hash and Verification: Through a series of OP_HASH160, OP_EQUALVERIFY, and another - OP_HASH160, the script performs cryptographic hash computations and verifications, ensuring the integrity and authenticity of the public key and secret hashes.
- OP_CHECKSIG: The final step involves checking the transaction's signature against the public key. If the signature is valid, the script confirms the transaction; otherwise, it fails.

Findings Breakdown

Critical	0
Medium	0
Minor / Informative	1

Severity	Unresolved	Acknowledged	Resolved	Other
Critical	0	0	0	0
Medium	0	0	0	0
Minor / Informative	1	0	0	0

Diagnostics

Critical Medium Minor / Informative

Severity	Code	Description	Status
	UTI	Unvalidated Transaction Inputs	Unresolved

UTI - Unvalidated Transaction Inputs

Criticality	Minor / Informative
Location	script
Status	Unresolved

Description

The script is designed in a manner where the success or failure of its execution is intrinsically linked to the inputs provided to the transaction. In this Bitcoin transaction script, the behavior and eventual outcome are depending upon the specifics of the unlocking script (or `scriptSig`) and any additional data that might be part of the transaction. This setup means that the script's ability to proceed correctly through its intended logic, including conditional paths and verification checks, is directly influenced by these initial inputs. If the inputs are incorrect, incomplete, or improperly formatted, the script will either fail to execute as intended or, more critically, could execute in an unintended manner, leading to potential vulnerabilities or transactional errors.

Recommendation

It is recommended to precisely validate the inputs provided to each transaction to ensure their correctness and completeness. This validation should include checks for proper formatting, authenticity of the signatures, and the accuracy of any public keys or secret hashes used. Implementing thorough input validation will safeguard the script from processing incorrect or unauthorized transactions. Such validation is not just a best practice but a critical security measure, especially in contexts where the script governs significant financial transactions or operates within a broader smart contract system. By ensuring that only correct and verified inputs are processed, the script can more reliably execute its intended logic, maintain transactional integrity, and uphold the security of the involved parties' assets.

Summary

The SmartSwap script is designed for secure Bitcoin transactions, employing conditional logic to facilitate complex escrow arrangements, with its effectiveness hinging critically on the accuracy and validation of input data.

Disclaimer

The information provided in this report does not constitute investment, financial or trading advice and you should not treat any of the document's content as such. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes nor may copies be delivered to any other person other than the Company without Cyberscope's prior written consent. This report is not nor should be considered an "endorsement" or "disapproval" of any particular project or team. This report is not nor should be regarded as an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Cyberscope to perform a security assessment. This document does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors' business, business model or legal compliance. This report should not be used in any way to make decisions around investment or involvement with any particular project. This report represents an extensive assessment process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk Cyberscope's position is that each company and individual are responsible for their own due diligence and continuous security Cyberscope's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies and in no way claims any guarantee of security or functionality of the technology we agree to analyze. The assessment services provided by Cyberscope are subject to dependencies and are under continuing development. You agree that your access and/or use including but not limited to any services reports and materials will be at your sole risk on an as-is where-is and as-available basis Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives false negatives and other unpredictable results. The services may access and depend upon multiple layers of third parties.

About Cyberscope

Cyberscope is a blockchain cybersecurity company that was founded with the vision to make web3.0 a safer place for investors and developers. Since its launch, it has worked with thousands of projects and is estimated to have secured tens of millions of investors' funds.

Cyberscope is one of the leading smart contract audit firms in the crypto space and has built a high-profile network of clients and partners.

The Cyberscope team

https://www.cyberscope.io