



# MatterLabs – Verifier

Smart Contract Security  
Assessment

Prepared by: Halborn

Date of Engagement: July 12th, 2023 – July 20th, 2023

Visit: [Halborn.com](https://Halborn.com)

DOCUMENT REVISION HISTORY	2
CONTACTS	2
1 EXECUTIVE OVERVIEW	3
1.1 INTRODUCTION	4
1.2 ASSESSMENT SUMMARY	4
1.3 SCOPE	5
1.4 TEST APPROACH & METHODOLOGY	6
2 RISK METHODOLOGY	7
2.1 EXPLOITABILITY	8
2.2 IMPACT	9
2.3 SEVERITY COEFFICIENT	11
3 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	13
4 MANUAL TESTING	13
5 AUTOMATED TESTING	24
5.1 STATIC ANALYSIS REPORT	25
Description	25
Slither results	25
5.2 AUTOMATED SECURITY SCAN	27
Description	27
MythX results	27

## DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Draft Document	07/21/2023	Gabi Urrutia
1.0	Final Report	09/15/2023	Gabi Urrutia

## CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	<a href="mailto:Rob.Behnke@halborn.com">Rob.Behnke@halborn.com</a>
Steven Walbroehl	Halborn	<a href="mailto:Steven.Walbroehl@halborn.com">Steven.Walbroehl@halborn.com</a>
Gabi Urrutia	Halborn	<a href="mailto:Gabi.Urrutia@halborn.com">Gabi.Urrutia@halborn.com</a>



# EXECUTIVE OVERVIEW



## 1.1 INTRODUCTION

This assessment was entirely focused on the new version of zkSync verifier which is a modified version of the Permutations over Lagrange-bases for Oecumenical Noninteractive arguments of Knowledge (PLONK) to optimize the proof system for zkSync Era circuits.

MatterLabs engaged Halborn to conduct a security assessment on their verifier smart contract beginning on July 12th, 2023 and ending on July 20th, 2023. The security assessment was scoped to the smart contract provided to the Halborn team.

## 1.2 ASSESSMENT SUMMARY

The team at Halborn was provided one week for the engagement and assigned a full-time security engineer to verify the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this assessment is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues within the smart contract

In summary, Halborn did not identify any security risks within the verifier smart contract.

## 1.3 SCOPE

### 1. IN-SCOPE:

The security assessment was scoped to the following [smart contract](#):

- [ethereum/contracts/verifier/Verifier.sol](#)

Commit ID: [f783f571e16a1b1adddb13db45db741f83b94812](#)

## 1.4 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this assessment. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the bridge code and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the assessment:

- Research into architecture and purpose.
- Smart contract manual code review and walkthrough.
- Graphing out functionality and contract logic/connectivity/functions. ([solgraph](#))
- Manual assessment of use and safety for the critical Solidity variables and functions in scope to identify any arithmetic related vulnerability classes.
- Manual testing by custom scripts.
- Scanning of solidity files for vulnerabilities, security hotspots or bugs. ([MythX](#))
- Static Analysis of security for scoped contract, and imported functions. ([Slither](#))
- Testnet deployment. ([Foundry](#))

## 2. RISK METHODOLOGY

Every vulnerability and issue observed by Halborn is ranked based on **two sets** of **Metrics** and a **Severity Coefficient**. This system is inspired by the industry standard Common Vulnerability Scoring System.

The two **Metric sets** are: **Exploitability** and **Impact**. **Exploitability** captures the ease and technical means by which vulnerabilities can be exploited and **Impact** describes the consequences of a successful exploit.

The **Severity Coefficients** is designed to further refine the accuracy of the ranking with two factors: **Reversibility** and **Scope**. These capture the impact of the vulnerability on the environment as well as the number of users and smart contracts affected.

The final score is a value between 0-10 rounded up to 1 decimal place and 10 corresponding to the highest security risk. This provides an objective and accurate rating of the severity of security vulnerabilities in smart contracts.

The system is designed to assist in identifying and prioritizing vulnerabilities based on their level of risk to address the most critical issues in a timely manner.



## 2.1 EXPLOITABILITY

### Attack Origin (AO):

Captures whether the attack requires compromising a specific account.

### Attack Cost (AC):

Captures the cost of exploiting the vulnerability incurred by the attacker relative to sending a single transaction on the relevant blockchain. Includes but is not limited to financial and computational cost.

### Attack Complexity (AX):

Describes the conditions beyond the attacker's control that must exist in order to exploit the vulnerability. Includes but is not limited to macro situation, available third-party liquidity and regulatory challenges.

### Metrics:

Exploitability Metric ( $m_E$ )	Metric Value	Numerical Value
Attack Origin (AO)	Arbitrary (AO:A)	1
	Specific (AO:S)	0.2
Attack Cost (AC)	Low (AC:L)	1
	Medium (AC:M)	0.67
	High (AC:H)	0.33
Attack Complexity (AX)	Low (AX:L)	1
	Medium (AX:M)	0.67
	High (AX:H)	0.33

Exploitability  $E$  is calculated using the following formula:

$$E = \prod m_e$$

## 2.2 IMPACT

### Confidentiality (C):

Measures the impact to the confidentiality of the information resources managed by the contract due to a successfully exploited vulnerability. Confidentiality refers to limiting access to authorized users only.

### Integrity (I):

Measures the impact to integrity of a successfully exploited vulnerability. Integrity refers to the trustworthiness and veracity of data stored and/or processed on-chain. Integrity impact directly affecting Deposit or Yield records is excluded.

### Availability (A):

Measures the impact to the availability of the impacted component resulting from a successfully exploited vulnerability. This metric refers to smart contract features and functionality, not state. Availability impact directly affecting Deposit or Yield is excluded.

### Deposit (D):

Measures the impact to the deposits made to the contract by either users or owners.

### Yield (Y):

Measures the impact to the yield generated by the contract for either users or owners.

## Metrics:

Impact Metric ( $m_I$ )	Metric Value	Numerical Value
Confidentiality (C)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Integrity (I)	None (I:N)	0
	Low (I:L)	0.25
	Medium (I:M)	0.5
	High (I:H)	0.75
	Critical (I:C)	1
Availability (A)	None (A:N)	0
	Low (A:L)	0.25
	Medium (A:M)	0.5
	High (A:H)	0.75
	Critical	1
Deposit (D)	None (D:N)	0
	Low (D:L)	0.25
	Medium (D:M)	0.5
	High (D:H)	0.75
	Critical (D:C)	1
Yield (Y)	None (Y:N)	0
	Low (Y:L)	0.25
	Medium: (Y:M)	0.5
	High: (Y:H)	0.75
	Critical (Y:H)	1

Impact  $I$  is calculated using the following formula:

$$I = \max(m_I) + \frac{\sum m_I - \max(m_I)}{4}$$

## 2.3 SEVERITY COEFFICIENT

### Reversibility (R):

Describes the share of the exploited vulnerability effects that can be reversed. For upgradeable contracts, assume the contract private key is available.

### Scope (S):

Captures whether a vulnerability in one vulnerable contract impacts resources in other contracts.

Coefficient ( $C$ )	Coefficient Value	Numerical Value
Reversibility ( $r$ )	None (R:N)	1
	Partial (R:P)	0.5
	Full (R:F)	0.25
Scope ( $s$ )	Changed (S:C)	1.25
	Unchanged (S:U)	1

Severity Coefficient  $C$  is obtained by the following product:

$$C = rs$$

The Vulnerability Severity Score  $S$  is obtained by:

$$S = \min(10, EIC * 10)$$

The score is rounded up to 1 decimal places.

Severity	Score Value Range
Critical	9 - 10
High	7 - 8.9
Medium	4.5 - 6.9
Low	2 - 4.4
Informational	0 - 1.9

### 3. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	0	0	0



# MANUAL TESTING



The main goal of the manual testing performed during this assessment was to test that the verifier is properly working to verify the zk proofs generated by the zkSync Era circuits, focusing on the following points/scenarios:

[illegible]



# MANUAL TESTING

Test	Result
Check that verifier reverts with proof is invalid if a maliciously forged serialized proof is sent	Pass

```

[Trace] Reason: loopProof (Proof is invalid) test_z2x2c() (gas: 133384)
[0x40973] jumpf()
  (0) word4 (170) [catalan]
    0xB0E9A5F70781481B30422475A836A4D3875E
  (1) word4 (170) [catalan]
    0x00000000000000000000000000000000
  (2) word4 (170) [catalan]
    0x1175 bytes of code
[0x40973] jumpf()
[133384] test_z2x2c()
  [8012] verifyf() verifyf(1238656575929848661564724329597608723927
[9237] 123555108495979933434369735124243,
[9238] 1235551084959799334369735124243,
[9239] 1235551084959799334369735124243,
[9240] 1235551084959799334369735124243,
[9241] 1235551084959799334369735124243,
[9242] 1235551084959799334369735124243,
[9243] 1235551084959799334369735124243,
[9244] 1235551084959799334369735124243,
[9245] 1235551084959799334369735124243,
[9246] 1235551084959799334369735124243,
[9247] 1235551084959799334369735124243,
[9248] 1235551084959799334369735124243,
[9249] 1235551084959799334369735124243,
[9250] 1235551084959799334369735124243,
[9251] 1235551084959799334369735124243,
[9252] 1235551084959799334369735124243,
[9253] 1235551084959799334369735124243,
[9254] 1235551084959799334369735124243,
[9255] 1235551084959799334369735124243,
[9256] 1235551084959799334369735124243,
[9257] 1235551084959799334369735124243,
[9258] 1235551084959799334369735124243,
[9259] 1235551084959799334369735124243,
[9260] 1235551084959799334369735124243,
[9261] 1235551084959799334369735124243,
[9262] 1235551084959799334369735124243,
[9263] 1235551084959799334369735124243,
[9264] 1235551084959799334369735124243,
[9265] 1235551084959799334369735124243,
[9266] 1235551084959799334369735124243,
[9267] 1235551084959799334369735124243,
[9268] 1235551084959799334369735124243,
[9269] 1235551084959799334369735124243,
[9270] 1235551084959799334369735124243,
[9271] 1235551084959799334369735124243,
[9272] 1235551084959799334369735124243,
[9273] 1235551084959799334369735124243,
[9274] 1235551084959799334369735124243,
[9275] 1235551084959799334369735124243,
[9276] 1235551084959799334369735124243,
[9277] 1235551084959799334369735124243,
[9278] 1235551084959799334369735124243,
[9279] 1235551084959799334369735124243,
[9280] 1235551084959799334369735124243,
[9281] 1235551084959799334369735124243,
[9282] 1235551084959799334369735124243,
[9283] 1235551084959799334369735124243,
[9284] 1235551084959799334369735124243,
[9285] 1235551084959799334369735124243,
[9286] 1235551084959799334369735124243,
[9287] 1235551084959799334369735124243,
[9288] 1235551084959799334369735124243,
[9289] 1235551084959799334369735124243,
[9290] 1235551084959799334369735124243,
[9291] 1235551084959799334369735124243,
[9292] 1235551084959799334369735124243,
[9293] 1235551084959799334369735124243,
[9294] 1235551084959799334369735124243,
[9295] 1235551084959799334369735124243,
[9296] 1235551084959799334369735124243,
[9297] 1235551084959799334369735124243,
[9298] 1235551084959799334369735124243,
[9299] 1235551084959799334369735124243,
[9300] 1235551084959799334369735124243,
[9301] 1235551084959799334369735124243,
[9302] 1235551084959799334369735124243,
[9303] 1235551084959799334369735124243,
[9304] 1235551084959799334369735124243,
[9305] 1235551084959799334369735124243,
[9306] 1235551084959799334369735124243,
[9307] 1235551084959799334369735124243,
[9308] 1235551084959799334369735124243,
[9309] 1235551084959799334369735124243,
[9310] 1235551084959799334369735124243,
[9311] 1235551084959799334369735124243,
[9312] 1235551084959799334369735124243,
[9313] 1235551084959799334369735124243,
[9314] 1235551084959799334369735124243,
[9315] 1235551084959799334369735124243,
[9316] 1235551084959799334369735124243,
[9317] 1235551084959799334369735124243,
[9318] 1235551084959799334369735124243,
[9319] 1235551084959799334369735124243,
[9320] 1235551084959799334369735124243,
[9321] 1235551084959799334369735124243,
[9322] 1235551084959799334369735124243,
[9323] 1235551084959799334369735124243,
[9324] 1235551084959799334369735124243,
[9325] 1235551084959799334369735124243,
[9326] 1235551084959799334369735124243,
[9327] 1235551084959799334369735124243,
[9328] 1235551084959799334369735124243,
[9329] 1235551084959799334369735124243,
[9330] 1235551084959799334369735124243,
[9331] 1235551084959799334369735124243,
[9332] 12355510849597993
```

Test	Result
Check that verifier reverts with proof is invalid if less than 44 words for serialized proof is sent	Pass

[illegible]

Test	Result
Check that verifier reverts with proof is invalid if less than 4 words for recursive aggregation input is sent	Pass

[illegible]

# MANUAL TESTING

Test	Result
Check that verifier returns a true for a public input with dirty bits over Fr mask	Pass

[illegible]

Test	Result
Check that the verifier returns a true having elliptic curve points over modulo	Pass

```
[
  (6980) PRECOMPILE::ecmd[1331481874871893334779197469607874446826148727324855612183283589578126862798, 13494143176161086751279221986335628979455862868313586262779787732264449104635, 544388980813155121095
[staticcall]
  14176619311897551523779426617675541851051754696965561673323857280623868374991, 942805240748525452694533683433685256822149357467517791521676738985137167172
[150] PRECOMPILE::ecmd[14176619311897551523779426617675541851051754696965561673323857280623868374991, 942805240748525452694533683433685256822149357467517791521676738985137167172, 4254236497062591530685
11994090[156851498312595815206597564868389791422122380174083048748483229546] [staticcall]
  81922286259707277820487314439987835740383113659220745992292767384820944, 14057338999727456585780628269294569308226029625678959458586547715828029287259
[6980] PRECOMPILE::ecmd[5558280385751162852868387327352787364885779562751792429916910431248322115, 1104992251481388972168491647526375723698288977736869249044998037679645888324, 16254379888580413159
[staticcall]
  1937196779573814775269798768338473832196883881343915638425221941727574348135, 1089649388375487236466545451076798374840893677525290438227227389697811734964
[150] PRECOMPILE::ecmd[1937196779573814775269798768338473832196883881343915638425221941727574348135, 1089649388375487236466545451076798374840893677525290438227227389697811734964, 816222286250976727378
14573383997274658897862625294569862212022625679894568485067715828029287259] [staticcall]
  19783988117207258939244146892242154715214074715817024729455598741524, 41294869707171988892968611665158313761385366225481923639672054919639522
[6980] PRECOMPILE::ecmd[10856868026458774927132263160471517755978915825982835215838391666972843, 858935648709842441004659165891219488880131785254312624515928607803222345483, 2859440854482125460292
[staticcall]
  43511261376574248258463399478453664678108608787741174697924489665768775869, 2149174878182179748286441203356136218320634541729553962942759848694863617
[150] PRECOMPILE::ecmd[43511261376574248258463399478453664678108608787741174697924489665768775869, 2149174878182179748286441203356136218320634541729553962942759848694863617, 178789088117267258035
12941869707171088892968611665158313761385366225481923639672054919639522] [staticcall]
  5210885641982981774897173835878080420824652264374347178445317795770194, 988803691962694293755104686276382217805637287880513695664431237740486939
[6980] PRECOMPILE::ecmd[22611083622087305555151428620446035983283181113150181569227811040171929804, 16576551425314335585292407242414297389775852406317563312380204893738838912, 791499858868497375234
[staticcall]
  2890748041386759274598898040679466951068434806831069948228634228568432604679, 901823855788213367405135187734467489611554681074667305571347762925130749570
[150] PRECOMPILE::ecmd[2890748041386759274598898040679466951068434806831069948228634228568432604679, 901823855788213367405135187734467489611554681074667305571347762925130749570, 36230885641982988179409
8863691962642939756104662768021780562722780851369566442327740486939] [staticcall]
  118293584061105747426254212328059782838476836397834683179891719937514227276, 175520379880745439163279773852558031436452583174449947783393969140693580439297
[150] PRECOMPILE::ecmd[118293584061105747426254212328059782838476836397834683179891719937514227276, 175520379880745439163279773852558031436452583174449947783393969140693580439297, 178528379880745439163279773852558031436452583174449947783393969140693580439297] [staticcall]
  1149543955805118217955804819808747812931792389316322844798163873074941466835, 181405894659459087268956824865255273155907906664975641879289804855104376660
[6980] PRECOMPILE::ecmd[1149543955805118217955804819808747812931792389316322844798163873074941466835, 181405894659459087268956824865255273155907906664975641879289804855104376660, 468973954140675352085
4689739541406753520852640769868507896381947521129204945477798459781705, 4811638130208526573732821419945255047045225639579132648796585807280881365
[150] PRECOMPILE::ecmd[1149543955805118217955804819808747812931792389316322844798163873074941466835, 181405894659459087268956824865255273155907906664975641879289804855104376660, 468973954140675352085
37788415098344868538476932027125382125658980920232688157786571526260718] [staticcall]
  47747746850845594613857740417398587864290529535104807245145008051232, 2394584229320218212015438046619555984116926553774137787692754087339451
[6980] PRECOMPILE::ecmd[16954586249736859288323125028096150913945599547384354941191381394834068, 77186637167944638637038235197981784115075292342754736299662633617841923857, 4106880893991154557899
[staticcall]
  201449646167923831807114083599642053713475374808519749776780798085487267749, 1274724232725892136957206257079899657258449411334678080412222563622414945252
[150] PRECOMPILE::ecmd[201449646167923831807114083599642053713475374808519749776780798085487267749, 1274724232725892136957206257079899657258449411334678080412222563622414945252, 477117740658845594613
2394584022983202121261643808166185589841169525637741377676092754807339451] [staticcall]
  6346892224664661340787423957330746522784627697513708020760720817484669, 151234871843646249355682968224985426586559218146334761807207706767459302904
[6980] PRECOMPILE::ecmd[6346892224664661340787423957330746522784627697513708020760720817484669, 151234871843646249355682968224985426586559218146334761807207706767459302904, 127262807007041743753592094979981820947420838770155024094245819658103406226, 190584558866807965610
[staticcall]
  1663818620873459504853384290675044200785415707646191904132945746435191188875, 75641180617729118984871892374985411079948681559558711543172308048398398
[150] PRECOMPILE::ecmd[1663818620873459504853384290675044200785415707646191904132945746435191188875, 75641180617729118984871892374985411079948681559558711543172308048398398, 51048572224663651340878
2348714756366249365652768212698425265865581816553474510920790867459302904] [staticcall]
  7692083708398059143239583949428469947331443272901714213789723833938, 172218547465923293808453294952466904466839277563406558470423995785041085929830
[6980] PRECOMPILE::ecmd[6346892224664661340787423957330746522784627697513708020760720817484669, 151234871843646249355682968224985426586559218146334761807207706767459302904, 127262807007041743753592094979981820947420838770155024094245819658103406226, 184754521677106392937
[staticcall]
  1123293802550457013487585789382643513513801783944807519406552349632414396, 965340646465527404516207104068236950517216152607570179281816395351116405359
[150] PRECOMPILE::ecmd[1123293802550457013487585789382643513513801783944807519406552349632414396, 965340646465527404516207104068236950517216152607570179281816395351116405359, 1695495862437338592832
718663721694453057828351197817841159752923476754369866233617841923807] [staticcall]
  8036632423471587731199966373780941668136485446805282996358948816212007376, 19181715804477795858612691356414522739525866120921154499188151555974440170007
[6980] PRECOMPILE::ecmd[1123293802550457013487585789382643513513801783944807519406552349632414396, 965340646465527404516207104068236950517216152607570179281816395351116405359, 1695495862437338592832
718663721694453057828351197817841159752923476754369866233617841923807] [staticcall]
  19377815587731199966373780941668136485446805282996358948816212007376, 19181715804477795858612691356414522739525866120921154499188151555974440170007
[6980] PRECOMPILE::ecmd[1123293802550457013487585789382643513513801783944807519406552349632414396, 965340646465527404516207104068236950517216152607570179281816395351116405359, 1695495862437338592832
718663721694453057828351197817841159752923476754369866233617841923807] [staticcall]
  705746978705642889027391718027467678371148908644421997881724674649451456, 10327563665084387605925182488211995438235473747342672865490738975834956
[150] PRECOMPILE::ecmd[705746978705642889027391718027467678371148908644421997881724674649451456, 10327563665084387605925182488211995438235473747342672865490738975834956, 5936632142347158738293
419581553812329266326704456475620798645941896179169381077156387921558079] [staticcall]
  155806486881215057504083967157404493006151922173135427606077249450120, 3677988212745994952772511641823214947140811749617621462411265802469723062121
[113080] PRECOMPILE::ecmd[155806486881215057504083967157404493006151922173135427606077249450120, 3677988212745994952772511641823214947140811749617621462411265802469723062121, 11059732082096
805634, 208578469992308571594457076222829481370756395785188699051999238565582781, 40823678758634368133220489145435568316851327693401208105741075214128093531, 849565839231234321417684973247489272438418190587263
  true
  console::log(true) [staticcall]
  ()
  ()
```

# MANUAL TESTING

Test	Result
Check that the verifier returns a true, having Fr over modulo	Pass

[illegible]

# MANUAL TESTING

Test	Result
Check that verifier reverts with proof is invalid if more than 1 public inputs is sent	Pass

[illegible]

Test	Result
Check that verifier reverts with proof is invalid if empty public inputs is sent	Pass

[illegible]

Test	Result
Check that verifier reverts with proof is invalid if more than 44 words for serialized proof is sent	Pass

[illegible]



Test	Result
Check that verifier reverts with proof is invalid if empty serialized proof is sent	Pass

```
[FAIL: Reason: loadProof: Proof is invalid] test_7x() (gas: 28877)
Traces:
[4648928] Pc::main()
├─ [0] VM::addr(170) [staticcall]
├─ [1] 0x0b9AF072C781481BF364224C75A036e4D832F52
├─ [2237390] = new Verifier0x0c7186504b17F316EC66f6e4422e1e82c47c246
├─ [1178 bytes of code]
└─ [0]
└─ [28877] Pc::test_7x()
├─ [6192] Verifier::verify([123865576579298458661564742432597860723927871731166708013954973021446496835], [1, [1257268268254973941446385474809939742774128922757452564483227089059504, 909121870191474859239
16188304989044819084910316745117567739259560693294530949925675821563, 3201893856796620567590585117572390872300853146726631773940175499357653053]]) [staticcall]
├─ [0] "loadProof: Proof is invalid"
└─ [0] "loadProof: Proof is invalid"
```

Test	Result
Check that verifier reverts with proof is invalid if more than 4 words for recursive aggregation input is sent	Pass

```
[FAIL: Reason: loadProof: Proof is invalid] test_3x3() (gas: 152188)
Traces:
[4648950] Pc::main()
├─ [0] VM::addr(170) [staticcall]
├─ [1] 0x0b9AF072C781481BF364224C75A036e4D832F52
├─ [2237390] = new Verifier0x0c7186504b17F316EC66f6e4422e1e82c47c246
├─ [1178 bytes of code]
└─ [0]
└─ [152188] Pc::test_3x3()
├─ [6102] Verifier::verify([123865576579298458661564742432597860723927871731166708013954973021446496835], [1967726114027632426084138238494384697720839663396536485256071746570736611109, 105129865931916946139204
7580924997258318844211365435687337212355106046959797334341858297351246243, 1695503689206246338512447293756017481444955756622473607731722999861633407461, 80488425405232386694105659521218478897410783052668321071772
894900745870530062325864919258585127, 12314818745719232477219746907074446563214897224056511032088097812662790, 1249414217616806761277221906325239745566106021350626777977226449194005, 230427644734053
900, 11111606123229434597473174598027378343951341138418065482707403415643711309103, 555820030675112828286839723752787364885777962751792429916910431240322115, 110499225148133897716849164752637572969280977378060
047151775597035150259820321538303166469878243, 850935640709043410845591658931940890813178525431262451592860700222345403, 479364195345958768741071519809469080393396373869583475157435610849387028, 211855663
63921767, 1836715149774686219439367764756076622824289237038995541864264440379305, 7522363890504916274532522461973214042409424712231943616857749008389799, 213944072633152048094933272001565986644
1385114207456504115407608462689274519301303386046004, 7981369842599851269275336294259594896287902316221967129915629062085969220, 19595044008418966439888866122313631646288170099502138673785162247912506528, 14
38458538224916730, 6460803985155618659464703419431451930481762187146772107908302116544934162, 609340472843681168727953902160000110677566736820501550623743948261491447128, 175705392746538515044166439416638593
825892089221678148743168085613307677856100138919240924, 1182293432536016448130676247602409773521199566438044466223095697090526, 6748304114833308561474606671389481487597859211192487914099409209
6949528243567412296814295, 11420964155381923928213509207160454711648431807177350760507962380804018828, 1077943715223456011086621945210636117389230424777974707626233879742468347341, 3454245479319341101497720211
12732913679347503933681770799747350008986778274765940152872241, 17066044038573449627374672600758088896497218881671974463428508685295680509, 1421676481241952736776144232258419527656794719447722848837087669
237813057624489599190844101161, 116625978164599647000114099356105826241062385236407230721820576414224137, 2190802115512922560151745812392426642399924696850400761526116399454046, 126539210113009373
1694958624763859208321831250280945891394559954737843649411013513941834068, 7718663721694453057608285119798178411507529234254736298406263617841923057, 631486125709382563947499341688471058627120690308085677674
947428838778105024974501956818346226], [2579286262549739414463854748099397427741289227577452564483227089059504, 9091218701914748532319691270014463917561734327715066129552313204917625530, 1618830498904
821563, 12089259579694656759056741767225907230085314673663177294017549935765305, 123865576579298458661564742432597860723927871731166708013954973021446496835]]) [staticcall]
├─ [0] "loadProof: Proof is invalid"
└─ [0] "loadProof: Proof is invalid"
```

Test	Result
Check that verifier reverts with proof is invalid if empty recursive aggregation input is sent	Pass

```
[FAIL: Reason: loadProof: Proof is invalid] test_8x() (gas: 116065)
Traces:
[4648950] Pc::main()
├─ [0] VM::addr(170) [staticcall]
├─ [1] 0x0b9AF072C781481BF364224C75A036e4D832F52
├─ [2237390] = new Verifier0x0c7186504b17F316EC66f6e4422e1e82c47c246
├─ [1178 bytes of code]
└─ [0]
└─ [116065] Pc::test_8x()
├─ [6102] Verifier::verify([123865576579298458661564742432597860723927871731166708013954973021446496835], [1967726114027632426084138238494384697720839663396536485256071746570736611109, 105129865931916946139204
7580924997258318844211365435687337212355106046959797334341858297351246243, 1695503689206246338512447293756017481444955756622473607731722999861633407461, 80488425405232386694105659521218478897410783052668321071772
894900745870530062325864919258585127, 12314818745719232477219746907074446563214897224056511032088097812662790, 1249414217616806761277221906325239745566106021350626777977226449194005, 230427644734053
900, 11111606123229434597473174598027378343951341138418065482707403415643711309103, 555820030675112828286839723752787364885777962751792429916910431240322115, 110499225148133897716849164752637572969280977378060
047151775597035150259820321538303166469878243, 850935640709043410845591658931940890813178525431262451592860700222345403, 479364195345958768741071519809469080393396373869583475157435610849387028, 211855663
63921767, 1836715149774686219439367764756076622824289237038995541864264440379305, 7522363890504916274532522461973214042409424712231943616857749008389799, 213944072633152048094933272001565986644
1385114207456504115407608462689274519301303386046004, 7981369842599851269275336294259594896287902316221967129915629062085969220, 19595044008418966439888866122313631646288170099502138673785162247912506528, 14
38458538224916730, 646080398515561865946470341943145193048177350760507962380804018828, 1077943715223456011086621945210636117389230424777974707626233879742468347341, 3454245479319341101497720211
12732913679347503933681770799747350008986778274765940152872241, 17066044038573449627374672600758088896497218881671974463428508685295680509, 1421676481241952736776144232258419527656794719447722848837087669
237813057624489599190844101161, 116625978164599647000114099356105826241062385236407230721820576414224137, 2190802115512922560151745812392426642399924696850400761526116399454046, 126539210113009373
1694958624763859208321831250280945891394559954737843649411013513941834068, 7718663721694453057608285119798178411507529234254736298406263617841923057, 631486125709382563947499341688471058627120690308085677674
947428838778105024974501956818346226], [1]) [staticcall]
├─ [0] "loadProof: Proof is invalid"
└─ [0] "loadProof: Proof is invalid"
```

Test	Result
Check that verifier reverts with proof is invalid if elliptic curve point at infinity is sent within the serialized proof	Pass

```
[FAIL: Reason: loadProof: Proof is invalid] test_2x2() (gas: 133267)
Traces:
[444958] Pc::setUp()
├─ (0) VM::idont(170) [staticcall]
├─ 0x0b9AF072C781481BF364224C76A0364A0832F52
├─ [2237390] = new Verifier(0x0C7186504017F316C666f4422e1e82c47c246
├─ = 11178 bytes of code
├─ ()
└─ [132667] Pc::test_2x2()
├─ [1818] Verifier::verify((112386557657929845866156472423259786723927871731166783813954973021444496256), [0, 0, 24788893469873883188443211365436487397212355184643697993434185539735124249, 1095886597206246338
1, 8848842485732866941565955212847889741878862668210717227938114984581969, 587691167418898494748557819297569818189438674578538863228586919255855157, 12314818748738334779154980767446682614877373485561
2897945628681386626779787732264494184835, 3284276642784083486687452786397872617861188802805938342486128635899980, 11116621222942459742174598882737834395131188418654827674031543171389183, 55532803305
4822115, 1184927518138971476237572968288077368087240449880787964588324, 1088368680245308745922122610484717557889158259928362158380912668072849, 88085824076994341188425910589319488880175826
98944088363399437898958474515747455418849287028, 2113850531224725524283998411819453138886292612231559247289261639271767, 18347316447776892319428977472684672228243859237859553418436446397395, 722
857769841897893, 212962487366238284889449327280156559586447155119684932475491425146738086, 2398778647796786578897138511420745565841154078684626897451938138388646804, 7981369842599514269276323242559548962
9888561223145611458170870898138876785186224791265066238, 14581629485284782728914938475378088981814438766221886487023848838234716736, 64688833985155618657464904194314518938481702187140771079888031154493416
6274943264149444728, 17578939239465385188416642914664389380749592128478131878423281244564, 1579467738992353288920622474783187481485863138374785618813891518934, 111822843252694548188476178
4833858561746486671889841748759678592113928479189948092897878, 142264765229469128196250472140624319859595246086949522823567412296814295, 1142096415538123912612892571664547164841180718775867685972389
2477974787621387974246847341, 34542454793193411011497728211518918186264825467245784543145602240417238972, 41288901297329163679247569783868177079974473888808678274776896812872241, 178648448387436492797436
47481241927378744432265419327547971347723848327204907286998938, 16131387246474458454546728385623182454537813877634049593980148181165, 1168298751645994780811480928554385324418628423618732072
66129936485584862562361769394945685, 126539191811020595773781298972851284198265211223707631453168443648459481, 1695498862437638892083221831250280961589139455954737843549411013513941834868, 77186627169445
97, 631486116708912659177499141884718686271286983888867767474268715882282429, 127262878078417477638720497798182094742883877818582489424819683184862261, 12257782682626497493944438847438979747274212892272
1446392765178429774159411959231328491756259, 16188344939844838189439883737519716872382597846982624538949286705821563, 328189355774962655798955117672598873288031474738637729484754987453851] [static
├─ "loadProof: Proof is invalid"
├─ "loadProof: Proof is invalid"
```

Test	Result
Check that the verifier reverts with invalid quotient evaluation if an invalid public input is used	Pass

```
[FAIL: Reason: invalid quotient evaluation] test_5x() (gas: 149561)
Traces:
[444958] Pc::setUp()
├─ (0) VM::idont(170) [staticcall]
├─ 0x0b9AF072C781481BF364224C76A0364A0832F52
├─ [2237390] = new Verifier(0x0C7186504017F316C666f4422e1e82c47c246
├─ = 11178 bytes of code
├─ ()
└─ [149561] Pc::test_5x()
├─ [27583] Verifier::verify((112386557657929845866156472423259786723927871731166783813954973021444496256), [196772611402962426884138238494384697280839643396536485268771764570736611109, 10511986593191694613920
47889344987358318844321136425687372123518664695729793343195827351246243, 16958638972862463851247293756717481444957566227436877317220999881632487461, 8848842485248523286694166595522128478897410738626682107187
1894389748378368632383689192585517, 123148187487383347791549807674466826148773734855612897945852084386262779787732244494184835, 3284276642784083486687452786397872617861188802805938342486128635899980, 11116621222942459742174598882737834395131188418654827674031543171389183, 55532803305
4822115, 1184927518138971476237572968288077368087240449880787964588324, 1088368680245308745922122610484717557889158259928362158380912668072849, 8808582407699434118842591058931948880175826
98944088363399437898958474515747455418849287028, 2113850531224725524283998411819453138886292612231559247289261639271767, 18347316447776892319428977472684672228243859237859553418436446397395, 722
857769841897893, 212962487366238284889449327280156559586447155119684932475491425146738086, 2398778647796786578897138511420745565841154078684626897451938138388646804, 7981369842599514269276323242559548962
9888561223145611458170870898138876785186224791265066238, 14581629485284782728914938475378088981814438766221886487023848838234716736, 64688833985155618657464904194314518938481702187140771079888031154493416
6274943264149444728, 17578939239465385188416642914664389380749592128478131878423281244564, 1579467738992353288920622474783187481485863138374785618813891518934, 111822843252694548188476178
4833858561746486671889841748759678592113928479189948092897878, 142264765229469128196250472140624319859595246086949522823567412296814295, 1142096415538123912612892571664547164841180718775867685972389
2477974787621387974246847341, 34542454793193411011497728211518918186264825467245784543145602240417238972, 41288901297329163679247569783868177079974473888808678274776896812872241, 178648448387436492797436
47481241927378744432265419327547971347723848327204907286998938, 16131387246474458454546728385623182454537813877634049593980148181165, 1168298751645994780811480928554385324418628423618732072
66129936485584862562361769394945685, 126539191811020595773781298972851284198265211223707631453168443648459481, 1695498862437638892083221831250280961589139455954737843549411013513941834868, 77186627169445
97, 63148611670891265917749914188471868627128698388867767474268715882282429, 127262878078417477638720497798182094742883877818582489424819683184862261, 12257782682626497493944438847438979747274212892272
1446392765178429774159411959231328491756259, 16188344939844838189439883737519716872382597846982624538949286705821563, 328189355774962655798955117672598873288031474738637729484754987453851] [static
├─ "invalid quotient evaluation"
├─ "invalid quotient evaluation"
```

Test	Result
Check that the verifier reverts with pairing failure if an invalid recursive aggregative input is used	Pass

[illegible]





# AUTOMATED TESTING



## 5.1 STATIC ANALYSIS REPORT

Description:

Halborn used automated testing techniques to enhance the coverage of certain areas of the scoped contract. Among the tools used was Slither, a Solidity static analysis framework. After Halborn verified the contract in the repository and was able to compile it correctly into their ABI and binary formats, Slither was run on the verifier contract. This tool can statically verify mathematical relationships between Solidity variables to detect invalid or inconsistent usage of the contracts' APIs across the entire code-base.

## Slither results:

# ethereumcontracts.verifier.Verifier.sol

```
Verifier.verifyOnChainHash((src/verifier/Verifier.sol#263-263) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#266-267)
Verifier.loadVerificationKey((src/verifier/Verifier.sol#276-324) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#277-323)
Verifier.verifyOnChainCommitment(commitment,(src/verifier/Verifier.sol#329-367) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#330-367b)
Verifier.verify_asm_0_revertWithMessage() (src/verifier/Verifier.sol#367-358) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#367-368)
Verifier.verify_asm_0_modexp() (src/verifier/Verifier.sol#361-372) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#361-372)
Verifier.verify_asm_0_pointMultIntTest() (src/verifier/Verifier.sol#375-382) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#375-382)
Verifier.verify_asm_0_pointAddIntTest() (src/verifier/Verifier.sol#383-393) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#383-393)
Verifier.verify_asm_0_pointSubAssign() (src/verifier/Verifier.sol#396-404) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#396-404)
Verifier.verify_asm_0_pointAddAssign() (src/verifier/Verifier.sol#407-415) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#407-415)
Verifier.verify_asm_0_pointMulModIntTest() (src/verifier/Verifier.sol#418-431) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#418-431)
Verifier.verify_asm_0_pointNegate() (src/verifier/Verifier.sol#434-445) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#434-445)
Verifier.verify_asm_0_updateTranscript() (src/verifier/Verifier.sol#452-460) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#452-460)
Verifier.verify_asm_0_permutationChallenge() (src/verifier/Verifier.sol#463-467) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#463-467)
Verifier.verify_asm_0_loadProof() (src/verifier/Verifier.sol#469-482) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#469-482)
Verifier.verify_asm_0_initializeTranscript() (src/verifier/Verifier.sol#478-794) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#478-794)
Verifier.verify_asm_0_verifyPointIntEvaluation() (src/verifier/Verifier.sol#818-846) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#818-846)
Verifier.verify_asm_0_evaluateLagrangePolynomialOverDomain() (src/verifier/Verifier.sol#865-882) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#865-882)
Verifier.verify_asm_0_permutationQuotientContribution() (src/verifier/Verifier.sol#885-943) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#885-943)
Verifier.verify_asm_0_lookupQuotientContribution() (src/verifier/Verifier.sol#946-988) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#946-988)
Verifier.verify_asm_0_verifyPermutationContribution() (src/verifier/Verifier.sol#991-1018) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#991-1018)
Verifier.verify_asm_0_addAssignmentCustomizedLinearizationContributionWithV() (src/verifier/Verifier.sol#1021-1048) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#1021-1048)
Verifier.verify_asm_0_verifyPointIntEvaluation() (src/verifier/Verifier.sol#1051-1127) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#1051-1127)
Verifier.verify_asm_0_prepareQuarrels() (src/verifier/Verifier.sol#1263-1339) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#1263-1339)
Verifier.verify_asm_0_findPairing() (src/verifier/Verifier.sol#1591-1644) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#1591-1644)
References: https://github.com/cryptic/alltherwiki/Docker-DocumentationAssembly-usage
INFO:Detectors:
Verifier.verify_asm_0_pointAddIntTest() (src/verifier/Verifier.sol#383-393) is never used and should be removed
INFO:Detectors:
https://github.com/cryptic/alltherwiki/Docker-DocumentationDead-code-detector
INFO:Detectors:
Prague version 0.8.13 (src/verifier/Verifier.sol#3) allows old versions
Pragma version 0.8.13 (src/verifier/Verifier.sol#3) allows old versions
Note 0.8.0-13 is not recommended for deployment
https://github.com/cryptic/alltherwiki/Docker-DocumentationIncorrect-versions-of-solidity
INFO:Detectors:
Parameter Verifier.verify_asm_0_revertWithMessage().len_verify_asm_0_revertWithMessage (src/verifier/Verifier.sol#367) is not in mixedCase
Parameter Verifier.verify_asm_0_revertWithMessage().msg_verify_asm_0_revertWithMessage (src/verifier/Verifier.sol#367) is not in mixedCase
Parameter Verifier.verify_asm_0_modexp().value_verify_asm_0_modexp (src/verifier/Verifier.sol#361) is not in mixedCase
Parameter Verifier.verify_asm_0_modexp().power_verify_asm_0_modexp (src/verifier/Verifier.sol#363) is not in mixedCase
Parameter Verifier.verify_asm_0_pointMultIntTest().x_verify_asm_0_pointMultIntTest (src/verifier/Verifier.sol#375) is not in mixedCase
Parameter Verifier.verify_asm_0_pointMultIntTest().y_verify_asm_0_pointMultIntTest (src/verifier/Verifier.sol#375) is not in mixedCase
Parameter Verifier.verify_asm_0_pointAddIntTest().dest_verify_asm_0_pointAddIntTest (src/verifier/Verifier.sol#383) is not in mixedCase
Parameter Verifier.verify_asm_0_pointAddIntTest().p2_verify_asm_0_pointAddIntTest (src/verifier/Verifier.sol#388) is not in mixedCase
Parameter Verifier.verify_asm_0_pointAddAssign().dest_verify_asm_0_pointAddAssign (src/verifier/Verifier.sol#404) is not in mixedCase
Parameter Verifier.verify_asm_0_pointAddAssign().p2_verify_asm_0_pointAddAssign (src/verifier/Verifier.sol#407) is not in mixedCase
Parameter Verifier.verify_asm_0_pointSubAssign().dest_verify_asm_0_pointSubAssign (src/verifier/Verifier.sol#415) is not in mixedCase
Parameter Verifier.verify_asm_0_pointSubAssign().p2_verify_asm_0_pointSubAssign (src/verifier/Verifier.sol#418) is not in mixedCase
Parameter Verifier.verify_asm_0_pointMulModIntTest().dest_verify_asm_0_pointMulModIntTest (src/verifier/Verifier.sol#418) is not in mixedCase
Parameter Verifier.verify_asm_0_pointMulModIntTest().p2_verify_asm_0_pointMulModIntTest (src/verifier/Verifier.sol#423) is not in mixedCase
Parameter Verifier.verify_asm_0_pointNegate().dest_verify_asm_0_pointNegate (src/verifier/Verifier.sol#434) is not in mixedCase
Parameter Verifier.verify_asm_0_pointNegate().p2_verify_asm_0_pointNegate (src/verifier/Verifier.sol#439) is not in mixedCase
Parameter Verifier.verify_asm_0_updateTranscript().dest_verify_asm_0_updateTranscript (src/verifier/Verifier.sol#452) is not in mixedCase
Parameter Verifier.verify_asm_0_updateTranscript().p2_verify_asm_0_updateTranscript (src/verifier/Verifier.sol#457) is not in mixedCase
Parameter Verifier.verify_asm_0_permutationChallenge().dest_verify_asm_0_permutationChallenge (src/verifier/Verifier.sol#463) is not in mixedCase
Parameter Verifier.verify_asm_0_permutationChallenge().p2_verify_asm_0_permutationChallenge (src/verifier/Verifier.sol#467) is not in mixedCase
Parameter Verifier.verify_asm_0_loadProof().dest_verify_asm_0_loadProof (src/verifier/Verifier.sol#469) is not in mixedCase
Parameter Verifier.verify_asm_0_loadProof().p2_verify_asm_0_loadProof (src/verifier/Verifier.sol#474) is not in mixedCase
Parameter Verifier.verify_asm_0_initializeTranscript().dest_verify_asm_0_initializeTranscript (src/verifier/Verifier.sol#478) is not in mixedCase
Parameter Verifier.verify_asm_0_initializeTranscript().p2_verify_asm_0_initializeTranscript (src/verifier/Verifier.sol#483) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPointIntEvaluation().dest_verify_asm_0_verifyPointIntEvaluation (src/verifier/Verifier.sol#818) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPointIntEvaluation().p2_verify_asm_0_verifyPointIntEvaluation (src/verifier/Verifier.sol#823) is not in mixedCase
Parameter Verifier.verify_asm_0_evaluateLagrangePolynomialOverDomain().dest_verify_asm_0_evaluateLagrangePolynomialOverDomain (src/verifier/Verifier.sol#865) is not in mixedCase
Parameter Verifier.verify_asm_0_evaluateLagrangePolynomialOverDomain().p2_verify_asm_0_evaluateLagrangePolynomialOverDomain (src/verifier/Verifier.sol#870) is not in mixedCase
Parameter Verifier.verify_asm_0_permutationQuotientContribution().dest_verify_asm_0_permutationQuotientContribution (src/verifier/Verifier.sol#885) is not in mixedCase
Parameter Verifier.verify_asm_0_permutationQuotientContribution().p2_verify_asm_0_permutationQuotientContribution (src/verifier/Verifier.sol#890) is not in mixedCase
Parameter Verifier.verify_asm_0_lookupQuotientContribution().dest_verify_asm_0_lookupQuotientContribution (src/verifier/Verifier.sol#946) is not in mixedCase
Parameter Verifier.verify_asm_0_lookupQuotientContribution().p2_verify_asm_0_lookupQuotientContribution (src/verifier/Verifier.sol#951) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPermutationContribution().dest_verify_asm_0_verifyPermutationContribution (src/verifier/Verifier.sol#991) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPermutationContribution().p2_verify_asm_0_verifyPermutationContribution (src/verifier/Verifier.sol#996) is not in mixedCase
Parameter Verifier.verify_asm_0_addAssignmentCustomizedLinearizationContributionWithV().dest_verify_asm_0_addAssignmentCustomizedLinearizationContributionWithV (src/verifier/Verifier.sol#1021) is not in mixedCase
Parameter Verifier.verify_asm_0_addAssignmentCustomizedLinearizationContributionWithV().p2_verify_asm_0_addAssignmentCustomizedLinearizationContributionWithV (src/verifier/Verifier.sol#1026) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPointIntEvaluation().dest_verify_asm_0_verifyPointIntEvaluation (src/verifier/Verifier.sol#1051) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPointIntEvaluation().p2_verify_asm_0_verifyPointIntEvaluation (src/verifier/Verifier.sol#1056) is not in mixedCase
Parameter Verifier.verify_asm_0_prepareQuarrels().dest_verify_asm_0_prepareQuarrels (src/verifier/Verifier.sol#1263) is not in mixedCase
Parameter Verifier.verify_asm_0_prepareQuarrels().p2_verify_asm_0_prepareQuarrels (src/verifier/Verifier.sol#1268) is not in mixedCase
Parameter Verifier.verify_asm_0_findPairing().dest_verify_asm_0_findPairing (src/verifier/Verifier.sol#1591) is not in mixedCase
Parameter Verifier.verify_asm_0_findPairing().p2_verify_asm_0_findPairing (src/verifier/Verifier.sol#1596) is not in mixedCase
```

[illegible]

- As a result of the tests carried out with the Slither tool, some results were obtained and reviewed by Halborn. Based on the results reviewed, the vulnerabilities were determined to be false positives.

## 5.2 AUTOMATED SECURITY SCAN

### Description:

Halborn used automated security scanners to assist with detection of well-known security issues, and to identify low-hanging fruits on the targets for this engagement. Among the tools used was MythX, a security analysis service for Ethereum smart contracts. MythX performed a scan on the verifier contract and sent the compiled results to the analyzers to locate any vulnerabilities.

### MythX results:

Report for Verifier.sol  
<https://dashboard.mythx.io/#/console/analyses/38dc0fe1-0e22-4008-a6a8-2c44e81bfcf>

Line	SWC Title	Severity	Short Description
3	(SWC-103) FloatingPragma	Low	A floating pragma is set.

- No major issues found by Mythx.



THANK YOU FOR CHOOSING

 **HALBORN**

