



MatterLabs – Verifier

Smart Contract Security
Assessment

Prepared by: Halborn

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

Visit: 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	Document Creation	07/19/2023	Omar Alshaeb
0.2	Document Updates	07/20/2023	Omar Alshaeb
0.3	Draft Review	07/21/2023	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Gokberk Gulgun	Halborn	Gokberk.Gulgun@halborn.com
Omar Alshaeb	Halborn	Omar.Alshaeb@halborn.com



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 assessment 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:

Test	Result
Check that using any valid proof, the verifier is able to properly verify it and returns a true as a result	Pass

[illegible]

MANUAL TESTING

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

[illegible]

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]

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]
  1417661931189755152377942661767554185105175469696556173223857280623868374991, 942805240748525452694533683433685256822149357167157791521676738985137167172
[150] PRECOMPILE::ecmd[1417661931189755152377942661767554185105175469696556173223857280623868374991, 942805240748525452694533683433685256822149357167157791521676738985137167172, 4254236497062591530685
11994090[15685149831259581520659756848638979142212238017408304576848229546] [staticcall]
  8192228625970727782048731443998785740363113659220745992292767384820944, 140573389997274565857806282692945693022602962567895945858657715828029287259
[6980] PRECOMPILE::ecmd[5558280385751162852868387327352787364885779562751792429916910431248322115, 110499225148138897216849164752637572369828897773686924904499803769645888324, 16254379888580413159
[staticcall]
  1937196779573814775269798768338473832196883881343915638425221941727574348135, 108964938837548723666654451076798374840893677525290438227227389697811734964
[150] PRECOMPILE::ecmd[1937196779573814775269798768338473832196883881343915638425221941727574348135, 108964938837548723666654451076798374840893677525290438227227389697811734964, 816222286250976727378
14573383997274658897862625294569862212022625679894568485067715828029287259] [staticcall]
  19783988117207258939244146892242154715214074715817024729455598741524, 412941696707171988892968611665158313761385366225481923639672054919639522
[6980] PRECOMPILE::ecmd[10856868026458774927132263160471517755978915825982835215838391666972843, 8589356487098424410046591658912194888801317852543126245155928607803222345483, 2859440854482125460292
[staticcall]
  43511261376574248258463399478453664678108608787774117469792448965768775869, 2149174878182179748286411203356136218320634517295539629427598486948463617
[150] PRECOMPILE::ecmd[43511261376574248258463399478453664678108608787774117469792448965768775869, 2149174878182179748286411203356136218320634517295539629427598486948463617, 178789088117267258035
12941669707171088892968611665158313761385366225481923639672054919639522] [staticcall]
  5210885641982981774897173835878080420824652264374347178445317795770194, 988803691962694293755104686276382217805637287880513695664431237740486939
[6980] PRECOMPILE::ecmd[226110836220837305555151428620446035983283181113150181569227811040171929804, 16576514235314335585292407242414297389775852406317563312380204893738838912, 791499858868497375234
[staticcall]
  28907480413867592745988980406794669510684348060831069948228634228568432604679, 901823855788213367405135187734467489611554681074667305571347762925130749570
[150] PRECOMPILE::ecmd[28907480413867592745988980406794669510684348060831069948228634228568432604679, 901823855788213367405135187734467489611554681074667305571347762925130749570, 36230885641982988179409
8863691962642939756104662768021780562722780851369566442327740486939] [staticcall]
  11829358406110574742625421231280572838476836397834683179891719937514227276, 175520379880745439163279773852558031436452583174449947783393969140693580439297
[150] PRECOMPILE::ecmd[11829358406110574742625421231280572838476836397834683179891719937514227276, 175520379880745439163279773852558031436452583174449947783393969140693580439297, 178528379880745439163279773852558031436452583174449947783393969140693580439297] [staticcall]
  114954395580511821795894819808747812931792389316322844798163873074941466835, 181405894659459087268956824865255273155907906864975641879289804855104376660
[6980] PRECOMPILE::ecmd[114954395580511821795894819808747812931792389316322844798163873074941466835, 181405894659459087268956824865255273155907906864975641879289804855104376660, 4689739541406753520552640769868507896381947623129920445477798459781705, 481163813002085235573732821419945255047045225639579132648796585807280881365
[150] PRECOMPILE::ecmd[114954395580511821795894819808747812931792389316322844798163873074941466835, 181405894659459087268956824865255273155907906864975641879289804855104376660, 468973954140675352055
37788415698344863853847693202712538215658980920232688157786571526260718] [staticcall]
  47747746850845594613857740417398578648290529535618408725415008051232, 2394584229320218212015438016619555984116926553774137707692754087339451
[6980] PRECOMPILE::ecmd[1695458624973685928832312502809615091394559954738435491191381394834068, 77186637167944638637038235197981784115075292342754736299602633617841923857, 4106880893991154557899
[staticcall]
  201449646167923831807141083599942053713475374808519749776780798085487267749, 127472423275892136957206257079899657258449411334678080412222563622414945252
[150] PRECOMPILE::ecmd[201449646167923831807141083599942053713475374808519749776780798085487267749, 127472423275892136957206257079899657258449411334678080412222563622414945252, 477117740658845594613
239458402298320121270164380816618558984116952563774137707692754807339451] [staticcall]
  636689222466466134078742395733074622784627697513708020760720817484669, 151234871784364249355682968224985420586559218146334761807207706767459302904
[6980] PRECOMPILE::ecmd[63148412570983256394749934168847105862712069038508567767494260715032202429, 127262807007041743753592094979981820947420838770155024094245819658103406226, 190584558868697965610
[staticcall]
  166381082387955940583384290675044200785415707646191904132945746435191188875, 7564118061772911898487189237498541107994661559555711543172308048398398
[150] PRECOMPILE::ecmd[166381082387955940583384290675044200785415707646191904132945746435191188875, 7564118061772911898487189237498541107994661559555711543172308048398398, 51048572224636651340878
234871475636242936562748226985425265855818165534745109207908647459302904] [staticcall]
  76920837083980591432395839494286599473314432729501714213789723833938, 1722185474659232938045329495246690446683927756360558420423995785041085929830
[6980] PRECOMPILE::ecmd[63148412570983256394749934168847105862712069038508567767494260715032202429, 127262807007041743753592094979981820947420838770155024094245819658103406226, 184754521677106392937
[staticcall]
  1123293802550457013487585789382634513513801783944807519404552349632414396, 96540464652674045162071040682369505172161526075707928181639535116406359
[150] PRECOMPILE::ecmd[1123293802550457013487585789382634513513801783944807519404552349632414396, 96540464652674045162071040682369505172161526075707928181639535116406359, 718663721694453057828251197817841159752923476754362986623361784192387] [staticcall]
  803663242347155731199966373780941668136485446805282996358948816212007376, 19181715504477795858612691356414522739525866120921154499189151555974440170007
[6980] PRECOMPILE::ecmd[1257928268254499394144638547389939742774128922725774525544832278896253584, 98912187019147485323319691720844639175617343297716506112955213204917562538, 1343513438079057386667
[staticcall]
  8024313428423045587357324947397957265252641644225676732722819768063465919, 644963482299354727796424548929871531513803512602077974052798585866553074
[150] PRECOMPILE::ecmd[8024313428423045587357324947397957265252641644225676732722819768063465919, 644963482299354727796424548929871531513803512602077974052798585866553074, 22185476592398045224952605946405292775368558420623995785081085929038] [staticcall]
  19377815577811199966373780941668136485446805282996358948816212007376, 19181715504477795858612691356414522739525866120921154499189151555974440170007
[6980] PRECOMPILE::ecmd[161883849890748438189493589373791976472882599566983024530949920676450021563, 328109355679696266675960513176732998723800831646738631772994017549907675385, 1343513438079057386667
[staticcall]
  70574697870564288902739171802746767871148908644421997881724674649451456, 103275366508438760495182488211995438235473747342672865490738975834956, 5936632142347158738293
[150] PRECOMPILE::ecmd[70574697870564288902739171802746767871148908644421997881724674649451456, 103275366508438760495182488211995438235473747342672865490738975834956, 5936632142347158738293
41958155301232926326704456475628798645941896179169381077156387921558077249450120, 367798821274599495277251164182321494714081179017621462411265802469723062121
[113080] PRECOMPILE::ecmd[19877831587741199966373780941668136485446805282996358959685316212007376, 19181715504477795858613691356414532739525866230921154499189151555994408170007, 11559732032096
805634, 20857846999230571594457076222829481370756395785188699051999238565582781, 40823678758634368133220403145435568316851327593401208105741075214128093531, 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]

MANUAL TESTING

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

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

[illegible]

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

```

11411 Reason: loadProof: Proof is invalid) (gas: 116065)
11412
11413 [L40499] Pc1::init(c)
11414
11415   (b) [b] := load(L70) [staticcall]
11416
11417   (c) 0x04d49f702c781461816436422ac75a63640832f82
11418
11419   (d) 22309
11420
11421   (e) new Verifier(c)(b1665d417f316c60d46f422e1e82c47c246
11422
11423     11175 bytes of code
11424
11425   (f)
11426
11427 [L10965] Pc1::ret(c)
11428
11429   ([8102] Verifier::new(c)(1123856575697298458661564742325976697392707131166736813954973821446496838), (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11430
11431     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11432
11433     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11434
11435     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11436
11437     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11438
11439     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11440
11441     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11442
11443     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11444
11445     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11446
11447     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11448
11449     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11450
11451     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11452
11453     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11454
11455     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11456
11457     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11458
11459     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11460
11461     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11462
11463     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11464
11465     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11466
11467     1123856575697298458661564742325976697392707131166736813954973821446496838, (1667726114429632426884182284943846972083396365482566771645736736611109, 10512986593191694433294
11468
11469     1123856575697298458661564742325976697392707131166736813954973821446496838, (16677261144296324268841822849438
```

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] PcU::setUp()
  └─ (0) VM::code(170) [staticcall]
    └─ 0x0b9AF072C781481BF364224C76A0364A0832F52
      └─ (2237390) = new Verifier(0x0C7186504017F316C666f4422e1e82c47c246
        └─ 11178 bytes of code
          └─ ()

[123667] PcU::test_2x2()
  └─ (1819) Verifier::verify((112386557657929845866156472423259786723927871731166783813954973021444496256), (0, 0, 24788893469873883188443211365436487397212355184643697993434185539785124249, 10958865897206244338
    1, 8848842485732866941565955212847889741878862668210717227938114984581969, 58769116741889849474855781942975498181894386745876538863228886919255855157, 12314818748738334779154980767446682614877373485641
    2897945628681386626779787732264494184835, 328427664278408348687452786397872617861188802805938342486128635899980, 11116621222942459742174598882737834395131188418654827674031543171389183, 55532803305
    4822115, 1184927518138971476752375729682880773680872404498807879645888324, 1088368680245380774929211226110847517657889158259928362158380912668072849, 88085824076994341188425910589319488880175826
    98944088363399437898924715167435518849287028, 2113850531224725524283998411819453138886292612231559247209261639271767, 18347316147776892319428977476268722282438592378589534188426446397895, 722
    857769841809839, 21296248736623818284889449327208156559866447155119684932475491425146738086, 23987786477967865788971385114207455658411540786846268974519381383886A680A, 79813698425995142692763232942559548962
    9888861223145611405381708989138876785186224791265066238, 1458162948528478272807149384753708889818144387662218864870238488383234716736, 64688833985155618657464904194314518938481702187140771079888831154493416
    62749432641494447238, 178789392394653851884166429146643893807495921284781318784232812445654, 157996377889623523889206221747831487434885863138374785618813891518934, 111822863252694548188476178
    48338585617464866718898417487596785921113928479189948092897878, 142264765229469128196250472140624319858595244608649522823567412296814295, 1142096415538123912681238925716645471164841180718775867685972889
    2477974787621387974246847341, 34542454793193411011497728211518939186264825467245784543145602240417238972, 412889681297329163679247569783868177079974473888888678274776696812872241, 178648448387436492797436
    47481241927378744432265419327547971347723848327204907286998928, 1613138723447445842486728385623182454537813877634849593980148181165, 11682987516549947808114809285543893624418628542618732072
    66129936485868462562361769394945685, 12653919181102059577378129897285128419826521122237076314531684436848459981, 1695498862437638892083221831256288961589139455954737842549411013513941834868, 77186637169445
    67, 63148611678981265917749914188471858627128698388886776747424687158822824292, 127262878078417477638720497798182094742883877818582489424819683184862261, 1225778626826249749944438847138979747274212892272
    14463927651784297761594119592313284917562539, 161888449398484381894398833787157168723825978469822624538949286705821563, 3281893557749626557598595117672598873288831474738637729484754987453851) [static
    call]
    └─ "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] PcU::setUp()
  └─ (0) VM::code(170) [staticcall]
    └─ 0x0b9AF072C781481BF364224C76A0364A0832F52
      └─ (2237390) = new Verifier(0x0C7186504017F316C666f4422e1e82c47c246
        └─ 11178 bytes of code
          └─ ()

[149561] PcU::test_5x()
  └─ (27583) Verifier::verify((112386557657929845866156472423259786723927871731166783813954973021444496256), (196772611402962426884138238494384697280839643396536485268771764570736611109, 10511986593191694613920
    478893449873583188443211365425687372123518664695729793343195827351246243, 16958638972862463851247239756877481444957566227436877317220999881632487461, 8848842485248523286694166695522128478897418738626682107187
    189438974837858686323836891925585517, 12314818748738334779154980767446682614877373485641, 58769116741889849474855781942975498181894386745876538863228886919255855157, 12314818748738334779154980767446682614877373485641
    9980, 11116621222942459742174598882737834395131188418654827674031543171389183, 555328033054822115, 1184927518138971476752375729682880773680872404498807879645888324, 1088368680245380774929211226110847517657889158259928362158380912668072849
    880858240769943411884259105893194888017582698944088363399437898924715167435518849287028, 2113850531224725524283998411819453138886292612231559247209261639271767, 18347316147776892319428977476268722282438592378589534188426446397895
    722857769841809839, 21296248736623818284889449327208156559866447155119684932475491425146738086, 23987786477967865788971385114207455658411540786846268974519381383886A680A, 79813698425995142692763232942559548962
    9888861223145611405381708989138876785186224791265066238, 1458162948528478272807149384753708889818144387662218864870238488383234716736, 64688833985155618657464904194314518938481702187140771079888831154493416
    62749432641494447238, 178789392394653851884166429146643893807495921284781318784232812445654, 157996377889623523889206221747831487434885863138374785618813891518934, 111822863252694548188476178
    48338585617464866718898417487596785921113928479189948092897878, 142264765229469128196250472140624319858595244608649522823567412296814295, 1142096415538123912681238925716645471164841180718775867685972889
    2477974787621387974246847341, 34542454793193411011497728211518939186264825467245784543145602240417238972, 41288968129732916367924756978386817707997447388888678274776696812872241, 178648448387436492797436
    47481241927378744432265419327547971347723848327204907286998928, 1613138723447445842486728385623182454537813877634849593980148181165, 11682987516549947808114809285543893624418628542618732072
    66129936485868462562361769394945685, 12653919181102059577378129897285128419826521122237076314531684436848459981, 1695498862437638892083221831256288961589139455954737842549411013513941834868, 77186637169445
    67, 6314861167898126591774991418847185862712869838886776747424687158822824292, 127262878078417477638720497798182094742883877818582489424819683184862261, 1225778626826249749944438847138979747274212892272
    14463927651784297761594119592313284917562539, 161888449398484381894398833787157168723825978469822624538949286705821563, 3281893557749626557598595117672598873288831474738637729484754987453851) [static
    call]
    └─ "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

```

    - 144176643918759152327942661767554185195175469696556167323857286623868374991, 942895248748425452694533563433695256822214935716715779152167073898513747172
[150] PRGCOMP111:rcmd1(14176643918759152327942661767554185195175469696556167323857286623868374991, 942895248748425452694533563433695256822214935716715779152167073898513747172, 4254236497862591538585
1199469971565851495814259312306577669683699914252122398474938947763432225640) [statcall1]
    - 810222862589767273782688731443968783573807681133659326748992292767384829944, 1457338399727465958780826292456938022602962567989549545856715828029387259
[600] PRGCOMP11:rcmd1(55582003805116102852886839723735787364885777956275179242916910421240322115, 11849972514813889721684916475263757296982897773686092498449983767645688824, 162542378803680413159
[statcall1]
    - 1937180779573814776426979076038047383218668388134391563842521941727574348135, 1809649388375487236664564451070798374848893677525298438227222309697811734964
[150] PRGCOMP11:rcmd1(1937180779573814776426979076038047383218668388134391563842521941727574348135, 1809649388375487236664564451070798374848893677525298438227222309697811734964, 81022286258976727378
1457338399727465958780826292456938022602962567989549545856715828029387259
[600] PRGCOMP11:rcmd1(1937180779573814776426979076038047383218668388134391563842521941727574348135, 1809649388375487236664564451070798374848893677525298438227222309697811734964, 81022286258976727378
1457338399727465958780826292456938022602962567989549545856715828029387259
[600] PRGCOMP11:rcmd1(185866802645807493291226316847151775597893158259283823158583916669872843, 850973564078994344100468165891794888861317852543126246518928687082232454849, 2059444054482125460292
[statcall1]
    - 428121319765744282584633994784536646781038678774176459792448986976075866, 214917487818217974882864412833561362183286345417295539629427598486948463617
129418697071718889292866114651583137413865662545459238299328553194939523) [statcall1]
    - 3623885641929891794971738358078686236266523647634127704453177956781974, 9088839519626942975510469627630217805637282780512695664412337740486939
[600] PRGCOMP11:rcmd1(22611084623838738555515142862844483598238310111315010156922781104071929084, 1657565142353142365858292407242414297389775852406317563312888204987376938912, 79149965868497375234
[statcall1]
    - 28907404138675927469889608676669518684348688318699482863422858432604679, 90183285578821236740513518773446748961155468107466730551247762935130749570, 3623885641902880179409
88039491926924937651040862763802178056732778051369564431237740486939) [statcall1]
    - 11893880461365747482654821832057788288468363679034683179891179927514227276, 17852037988074543910277377852558031436042528314744994478339399914869350432927
[150] PRGCOMP11:rcmd1(12178698153194874494585604919072459776074189242448422868264160875841695831, 1281734062947653779264820891111921891544240737880100930256880820836877871941, 118393504061365747482
17852037988074543910277377852558031436042528314744994478339399914869350432927) [statcall1]
    - 11495438555810217965880410889847812931763897313228047781638730749414668635, 181405874659459807284895682486552527315590790664975641879289004055104376660
[600] PRGCOMP11:rcmd1(1, 2, 51899961689715451971399671105862637123286608188652405794692111668752802) [statcall1]
    - 469773951140675825895324076964685709630194785112994846477984159981785, 1811028158288526577133531419745625584706525639579132668796058837380081865
[150] PRGCOMP11:rcmd1(11495438555810217965880410889847812931763897313228047781638730749414668635, 181405874659459807284895682486552527315590790664975641879289004055104376660, 46977395114067582585
37780415098246648587682625712328626389980926923680818778507152366718) [statcall1]
    - 472177406583455944138572404177395878042985858851940872514598880612332, 2394954022982102182120143081661855589411695265637741377076892754807339461
[600] PRGCOMP11:rcmd1(14954385624763857208321831250280401509139455995473784354941101351391482468, 7718663721694453853708382351197817841150875292342754736278068263617841923857, 41186880933911554557099
[statcall1]
    - 20149466167922831307114105899462953713475374800591879776798979805407267749, 127472423725892136957204257075899865725844496113346784080412222536622414945252, 477117740658845594613
[150] PRGCOMP11:rcmd1(20149466167922831307114105899462953713475374800591879776798979805407267749, 127472423725892136957204257075899865725844496113346784080412222536622414945252, 477117740658845594613
2394954022982102182120143081661855589411695265637741377076892754807339461) [statcall1]
    - 510489222464665140837874529573307486278462769876137889287407200817406669, 51224871478366249365682968226985425080559818165534765109209780657459302904
[600] PRGCOMP11:rcmd1(43148612578932563947479341688471058627123869385885677674944260715082282429, 127262807804147437585209497998182094743208387701550240894245019658103406226, 198504556806607965610
[statcall1]
    - 1663810629379659048338442900760429007604510706619190421395657648355101188875, 756641108617729118984871892374985411079946681559555871154313720050483389398, 51048922246466514083787
[150] PRGCOMP11:rcmd1(1663810629379659048338442900760429007604510706619190421395657648355101188875, 756641108617729118984871892374985411079946681559555871154313720050483389398, 51048922246466514083787
2394954022982102182120143081661855589411695265637741377076892754807339461) [statcall1]
    - 78602803780259808051943295083949842869094733114937229581143137897238389398, 1722185476592253988453294952605904606392776436055828623995785601885929030
[600] PRGCOMP11:rcmd1(43148612578932563947479341688471058627123869385885677674944260715082282429, 127262807804147437585209497998182094743208387701550240894245019658103406226, 104754521677166392937
[statcall1]
    - 112319380265891578134875857893826363513138017889440975519406552369432414306, 9653406496525749495162571968682368055517316152607579179381816395635116605259
[150] PRGCOMP11:rcmd1(112319380265891578134875857893826363513138017889440975519406552369432414306, 9653406496525749495162571968682368055517316152607579179381816395635116605259, 16954958624376385920832
710663721694530697803251497081741510975302242547263868623884704923857) [statcall1]
    - 9366321123415087829318544858983270769131800836305167473857692959754455954, 184686613182803592561278559989918884789847138336831968878560788257384558504
[600] PRGCOMP11:rcmd1(1, 2, 4343513480790573866737210468516897188629913057797065793486745276687235395928) [statcall1]
    - 2104022336338026979673695480624292494366191429017528444867399469785452829, 2522446783643694680920354409497834611576643778942440101495757847187427771787
[150] PRGCOMP11:rcmd1(2104022336338026979673695480624292494366191429017528444867399469785452829, 2522446783643694680920354409497834611576643778942440101495757847187427771787
7221854765932598848522495260598494085392775636055862862395736561895529830) [statcall1]
    - 2104022336338026979673695480624292494366191429017528444867399469785452829, 2522446783643694680920354409497834611576643778842440101495757847187427771787
[150] PRGCOMP11:rcmd1(1, 2, 4343513480790573866737210468516897188629913057797065793486745276687235395928) [statcall1]
    - 2104022336338026979673695480624292494366191429017528444867399469785452829, 2522446783643694680920354409497834611576643778842440101495757847187427771787
[150] PRGCOMP11:rcmd1(2104022336338026979673695480624292494366191429017528444867399469785452829, 2522446783643694680920354409497834611576643778842440101495757847187427771787
41958105301237296336201467435620599446418961791693810471847892731058079) [statcall1]
    - 116088695745327780367573395844276591667738938846236254986427546426219095, 5346441151310719597073132137152726866316163249756659984309561676290887526
[113000] PRGCOMP11:rcmd1(12045335683839983394235241368076804100508280801664225427233813208188469975, 55839429285226039439364949574192363477540774065744762712470323444973083964, 115597320320863
95634, 1085704099923057194545707622232948137075435957831808499019999285455852781, 40823677586343268132220340314543568316851327593401208105741076214112009351, 8495653923123431471604973247829724384181905872636
= 0
- "finalPairing: pairing failure"
- "finalPairing: pairing failure"
```




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#718-794) uses assembly
    - INLINE ASM (src/verifier/Verifier.sol#718-794)
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/Verifer.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#347) is not in mixedCase
Parameter Verifier.verify_asm_0_revertWithMessage().msg_verify_asm_0_revertWithMessage (src/verifier/Verifier.sol#347) 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#375) is not in mixedCase
Parameter Verifier.verify_asm_0_pointAddIntTest().x_verify_asm_0_pointAddIntTest (src/verifier/Verifier.sol#383) is not in mixedCase
Parameter Verifier.verify_asm_0_pointAddIntTest().y_verify_asm_0_pointAddIntTest (src/verifier/Verifier.sol#383) is not in mixedCase
Parameter Verifier.verify_asm_0_pointSubAssign().p2_verify_asm_0_pointSubAssign (src/verifier/Verifier.sol#396) is not in mixedCase
Parameter Verifier.verify_asm_0_pointSubAssign().q2_verify_asm_0_pointSubAssign (src/verifier/Verifier.sol#396) 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_pointAddAssign().q2_verify_asm_0_pointAddAssign (src/verifier/Verifier.sol#407) is not in mixedCase
Parameter Verifier.verify_asm_0_pointMulModIntTest().p2_verify_asm_0_pointMulModIntTest (src/verifier/Verifier.sol#418) is not in mixedCase
Parameter Verifier.verify_asm_0_pointMulModIntTest().q2_verify_asm_0_pointMulModIntTest (src/verifier/Verifier.sol#418) is not in mixedCase
Parameter Verifier.verify_asm_0_pointNegate().p2_verify_asm_0_pointNegate (src/verifier/Verifier.sol#434) is not in mixedCase
Parameter Verifier.verify_asm_0_pointNegate().q2_verify_asm_0_pointNegate (src/verifier/Verifier.sol#434) is not in mixedCase
Parameter Verifier.verify_asm_0_updateTranscript().p2_verify_asm_0_updateTranscript (src/verifier/Verifier.sol#407) is not in mixedCase
Parameter Verifier.verify_asm_0_updateTranscript().q2_verify_asm_0_updateTranscript (src/verifier/Verifier.sol#407) is not in mixedCase
Parameter Verifier.verify_asm_0_permutationChallenge().p2_verify_asm_0_permutationChallenge (src/verifier/Verifier.sol#463) is not in mixedCase
Parameter Verifier.verify_asm_0_permutationChallenge().q2_verify_asm_0_permutationChallenge (src/verifier/Verifier.sol#463) is not in mixedCase
Parameter Verifier.verify_asm_0_loadProof().p2_verify_asm_0_loadProof (src/verifier/Verifier.sol#469) is not in mixedCase
Parameter Verifier.verify_asm_0_loadProof().q2_verify_asm_0_loadProof (src/verifier/Verifier.sol#469) is not in mixedCase
Parameter Verifier.verify_asm_0_initializeTranscript().p2_verify_asm_0_initializeTranscript (src/verifier/Verifier.sol#718) is not in mixedCase
Parameter Verifier.verify_asm_0_initializeTranscript().q2_verify_asm_0_initializeTranscript (src/verifier/Verifier.sol#718) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPointIntEvaluation().p2_verify_asm_0_verifyPointIntEvaluation (src/verifier/Verifier.sol#818) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPointIntEvaluation().q2_verify_asm_0_verifyPointIntEvaluation (src/verifier/Verifier.sol#818) is not in mixedCase
Parameter Verifier.verify_asm_0_evaluateLagrangePolynomialOverDomain().p2_verify_asm_0_evaluateLagrangePolynomialOverDomain (src/verifier/Verifier.sol#865) is not in mixedCase
Parameter Verifier.verify_asm_0_evaluateLagrangePolynomialOverDomain().q2_verify_asm_0_evaluateLagrangePolynomialOverDomain (src/verifier/Verifier.sol#865) is not in mixedCase
Parameter Verifier.verify_asm_0_permutationQuotientContribution().p2_verify_asm_0_permutationQuotientContribution (src/verifier/Verifier.sol#885) is not in mixedCase
Parameter Verifier.verify_asm_0_permutationQuotientContribution().q2_verify_asm_0_permutationQuotientContribution (src/verifier/Verifier.sol#885) is not in mixedCase
Parameter Verifier.verify_asm_0_lookupQuotientContribution().p2_verify_asm_0_lookupQuotientContribution (src/verifier/Verifier.sol#946) is not in mixedCase
Parameter Verifier.verify_asm_0_lookupQuotientContribution().q2_verify_asm_0_lookupQuotientContribution (src/verifier/Verifier.sol#946) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPermutationContribution().p2_verify_asm_0_verifyPermutationContribution (src/verifier/Verifier.sol#991) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPermutationContribution().q2_verify_asm_0_verifyPermutationContribution (src/verifier/Verifier.sol#991) is not in mixedCase
Parameter Verifier.verify_asm_0_addAssignmentCustomizedLinearizationContributionWithV().p2_verify_asm_0_addAssignmentCustomizedLinearizationContributionWithV (src/verifier/Verifier.sol#1021) is not in mixedCase
Parameter Verifier.verify_asm_0_addAssignmentCustomizedLinearizationContributionWithV().q2_verify_asm_0_addAssignmentCustomizedLinearizationContributionWithV (src/verifier/Verifier.sol#1021) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPointIntEvaluation().p2_verify_asm_0_verifyPointIntEvaluation (src/verifier/Verifier.sol#1051) is not in mixedCase
Parameter Verifier.verify_asm_0_verifyPointIntEvaluation().q2_verify_asm_0_verifyPointIntEvaluation (src/verifier/Verifier.sol#1051) is not in mixedCase
Parameter Verifier.verify_asm_0_prepareQuarrels().p2_verify_asm_0_prepareQuarrels (src/verifier/Verifier.sol#1263) is not in mixedCase
Parameter Verifier.verify_asm_0_prepareQuarrels().q2_verify_asm_0_prepareQuarrels (src/verifier/Verifier.sol#1263) is not in mixedCase
Parameter Verifier.verify_asm_0_findPairing().p2_verify_asm_0_findPairing (src/verifier/Verifier.sol#1591) is not in mixedCase
Parameter Verifier.verify_asm_0_findPairing().q2_verify_asm_0_findPairing (src/verifier/Verifier.sol#1591) 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**

