

# 阿凡达 (Avatar)

智能合约审计报告



编写者：

块状审计 (BlockAudit)

入学日期。  
2023年2月2日-2023年2月5日

访问：[www.blockaudit.report](http://www.blockaudit.report)



# 目录

<b>简介</b>	<b>2-3</b>
└ 摘要	2
└ 概述	3
<b>研究结果</b>	<b>4-10</b>
└ 搜索结果概述	4
└ BKTP01	5
└ BKTP02	7
└ BKTP03	8
└ BKTP04	10
<b>附录</b>	<b>11</b>
<b>免责声明</b>	<b>13</b>
<b>关于</b>	<b>15</b>





## 摘要

这份审计报告主要是针对Avatar智能合约的广泛安全性。通过这份报告，我们试图通过对系统架构和智能合约代码库的完整而严格的评估来确保智能合约的可靠性和正确性。

审计过程中特别注意以下几点考虑。

审计过程中特别注意以下几点考虑。

针对常见和不常见的攻击载体测试智能合约。

评估代码库以确保符合当前的最佳实践和行业标准。

确保合同逻辑符合客户的规格和意图。

将合同结构和执行情况与行业领导者制作的类似智能合同进行交叉参考。

由行业专家对整个代码库进行彻底的逐行人工审查。

安全评估的结果是，从关键到信息不等的发现。我们建议解决这些发现，以确保高水平的安全标准和行业惯例。我们提出了一些建议，可以从安全角度更好地服务于项目。

加强一般的编码实践，使源代码的结构更加完善。

增加足够的单元测试以覆盖可能的使用案例。

为每个功能提供更多的注释以提高可读性，尤其是公开验证的合同。

一旦协议生效，提供更多关于特权活动的透明度。



# 概述

## 项目摘要

项目名称	阿凡达 (Avatar)
语言	固体
平台	蓼蓝 (Polygon)
合同地址	0xC7728354f9fe0e43514B1227162D5B0E40FaD410

## 文件摘要

身份证	文件名称
BKTB	BKTO-Bucket.sol
BKTA	BKTO-Avatar.sol
BKTP	BKTO-PaymentSplitter.sol

交付日期	2023年2月5日
审计方法	代码分析。自动评估，人工审查
审计结果	通过 ✅
审计小组	BlockAudit报告团队



## 研究结果



### 脆弱性研究结果摘要

ID	类型	实例	严重程度	状况
BKTP01	Unlocked Pragma Used In Contract	55	低	认可
BKTP02	使用自定义错误而不是revert() / require() 字符串来节省气体。	1	低	已解决
BKTP03	在for-each循环中不必要的LOADs 和MLOs	6	低	已解决
BKTP04	TODO	1	低	已解决



# BKTP01

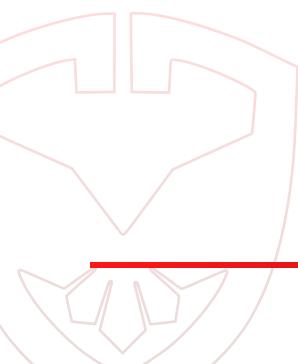
类型	使用自定义错误而不是revert() / require() 字符串来节省气体。
严重程度	■ 低
文件	Paymentsplitter.sol, Avatar.sol
实例	55
状况	认可

## 描述

自定义错误从solidity 0.8.4版本开始可用。自定义错误通过避免分配和存储还原字符串，每次击中都能节省约50个气体。不定义字符串也可以节省部署气体。

## 快照

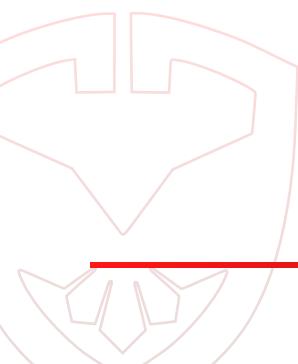
```
./solidity/PaymentSplitter.sol:56:      require(msg.sender == tempAdmin, "PaymentSplitter: only temp
admin");
./solidity/PaymentSplitter.sol:68:      require(msg.sender == tempAdmin, "PaymentSplitter: only temp
admin");
./solidity/PaymentSplitter.sol:69:      require(payees.length == shares_.length, "PaymentSplitter:
payees and shares length mismatch");
./solidity/PaymentSplitter.sol:70:      require(payees.length > 0, "PaymentSplitter: no payees");
./solidity/PaymentSplitter.sol:165:      require(_shares[account] > 0, "PaymentSplitter: account has
no shares");
./solidity/PaymentSplitter.sol:169:      require(payment != 0, "PaymentSplitter: account is not due
payment");
./solidity/PaymentSplitter.sol:188:      require(_shares[account] > 0, "PaymentSplitter: account has
no shares");
./solidity/PaymentSplitter.sol:192:      require(payment != 0, "PaymentSplitter: account is not due
payment");
./solidity/PaymentSplitter.sol:224:      require(account != address(0), "PaymentSplitter: account is
the zero address");
./solidity/PaymentSplitter.sol:225:      require(shares_ > 0, "PaymentSplitter: shares are 0");
./solidity/PaymentSplitter.sol:226:      require(_shares[account] == 0, "PaymentSplitter: account
already has shares");
```





## 快照

```
./solidity/Avatar.sol:171: require(msg.sender == tx.origin, "Contract not allowed");
./solidity/Avatar.sol:185: require(
./solidity/Avatar.sol:207:     msg.sender == operator, "Only operator");
./solidity/Avatar.sol:210:         require(tempAdmin == address(0), "Temp admin not dropped");
./solidity/Avatar.sol:219:         require(operator != address(0), "Invalid address");
./solidity/Avatar.sol:220:         require(msg.sender == tempAdmin, "Only admin");
./solidity/Avatar.sol:228:         require(msg.sender == tempAdmin, "Only admin");
./solidity/Avatar.sol:243:         require(users.length == referrers.length && users.length ==
salesLevels.length, "Invalid input");
./solidity/Avatar.sol:247:         require(users[i] != address(0), "Invalid address provided");
./solidity/Avatar.sol:249:         require(userGlobalInfo.referrer == address(0), "Referrer already
set");
./solidity/Avatar.sol:267:         require(ledgerType > 0, "Invalid ledger type");
./solidity/Avatar.sol:268:         require(ledgerType < 6, "Invalid ledger type");
./solidity/Avatar.sol:269:         require(msg.sender == tempAdmin, "Only admin");
./solidity/Avatar.sol:270:         require(stock.length > 0, "Invalid stock array");
./solidity/Avatar.sol:271:         require(typeDays.length == stock.length, "Invalid params");
./solidity/Avatar.sol:291:         require(ledgerType < 6, "Invalid ledger type");
./solidity/Avatar.sol:292:         require(targetEpoch == currentEpochs[ledgerType], "Invalid epoch");
./solidity/Avatar.sol:293:         require(msg.value >= MIN_INVEST, "Too small");
./solidity/Avatar.sol:294:         require(msg.value <= MAX_INVEST, "Too large");
./solidity/Avatar.sol:295:         require(!gamePaused, "Paused");
./solidity/Avatar.sol:310:             require(referrer != address(0) && referrer != msg.sender,
require(
./solidity/Avatar.sol:312:             require(targetRate <= params.investReturnRate, "Invalid ratio");
./solidity/Avatar.sol:334:                 require(boostCredit >= msg.value, "Exceed boost credit");
./solidity/Avatar.sol:366:                     require(success, "Transfer failed.");
./solidity/Avatar.sol:443:                     require(ledgerType < 6, "Invalid ledger type");
./solidity/Avatar.sol:460:                     require(epoch <= currentEpochs[ledgerType], "Invalid epoch");
./solidity/Avatar.sol:461:                     require(positionIndex < positionInfos.length, "Invalid position
index");
./solidity/Avatar.sol:465:                     require(ledgerType < 6, "Invalid ledger type");
./solidity/Avatar.sol:490:                     require(epoch <= currentEpochs[ledgerType], "Invalid epoch");
./solidity/Avatar.sol:491:                     require(positionIndexes.length > 0, "Invalid position indexes");
./solidity/Avatar.sol:492:                         require(positionIndexes[t] < positionInfos.length, "Invalid
position index");
./solidity/Avatar.sol:507:                         require(ledgerType < 6, "Invalid ledger type");
./solidity/Avatar.sol:525:                         require(epoch < currentEpochs[ledgerType], "Epoch not finished");
./solidity/Avatar.sol:526:                             require(positionIndex < positionInfos.length, "Invalid position
index");
./solidity/Avatar.sol:546:                             require(positionInfo.incentiveClaimable, "Position not
eligible");
./solidity/Avatar.sol:549:                             require(success, "Transfer failed.");
./solidity/Avatar.sol:557:                             require(referrer != address(0), "Invalid referrer address");
./solidity/Avatar.sol:580:                             require(claimableAmount > 0, "No claimable amount");
./solidity/Avatar.sol:588:                                 require(success, "Transfer failed.");
./solidity/Avatar.sol:596:                                 require(positionInfo.withdrawnAmount == 0, "Position already
claimed");
./solidity/Avatar.sol:727:             require(positionInfo.expiryTime <= block.timestamp ||
roundInfo.stopLoss, "Position not expired");
./solidity/Avatar.sol:822:                 require(success, "Transfer failed.");
```





## BKTP02

类型	合同中使用的解锁pragma
严重程度	■ 低
文件	PaymentSplitter.sol (付款分流器)。
实例	1
状况	已解决

### 描述

大多数合同使用一个未锁定的pragma（例如，pragma solidity ^0.8.0），它不固定在特定的Solidity版本上。锁定 pragma 有助于确保合同不会意外地使用不同的编译器版本进行部署，而这些编译器版本已经过最多的测试。请使用 grep -R pragma . 来查找代码库中未锁定的pragma语句

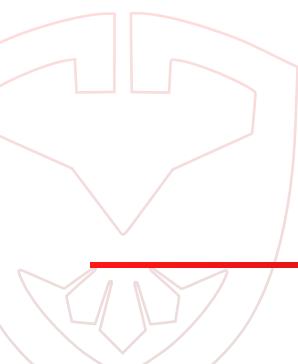
### 补救措施

建议将 pragmas 锁定为特定的 Solidity 版本。考虑以下列表中的编译器错误，并确保合约不受它们的影响。在部署合约时，也建议使用最新版本的 Solidity（见 Solidity docs）。

### 快照

The screenshot shows a terminal window with a black background and white text. In the top left corner, there are three small colored circles (red, yellow, green). The text in the terminal is:

```
./solidity/PaymentSplitter.sol:4:pragma solidity  
^0.8.0;
```





## BKTP03

类型	在for-each循环中不必要的LOADs和MLOs
严重程度	低
文件	所有文件
实例	6
状况	已解决

### 描述

有很多for循环都遵循这种for-each模式。

```
● ● ●  
for (uint256 i = 0; i < array.length; i++)  
{ // do something with `array[i]`  
}
```

在这样的for循环中，array.length在每次迭代中都被读取，而不是在一个局部变量中缓存一次，然后从那里读取。存储器的读取比读取局部变量要贵得多。内存读取比读取局部变量要贵一些。

### 补救措施

```
● ● ●  
uint256 length = array.length;  
for (uint256 i = 0; i < length; i++)  
{ // do something with `array[i]`  
}
```



## 快照

A screenshot of a terminal window with a dark background. It displays several lines of Solidity code. The code includes imports from 'solidity/PaymentSplitter.sol', 'solidity/Avatar.sol', and 'solidity/Bucket.sol'. It contains multiple nested loops using 'for' statements with conditions like 'i < payees.length' and 'i < users.length'.

```
./solidity/PaymentSplitter.sol:72:      for (uint256 i = 0; i < payees.length; i++) {  
./solidity/Avatar.sol:246:      for (uint256 i = 0; i < users.length; ++i) {  
./solidity/Avatar.sol:506:      for (uint256 i = 0; i < positionIndexes.length; ++i)  
{/solidity/Avatar.sol:544:      for (uint256 i = 0; i < positionIndexes.length; ++i)  
{/solidity/Avatar.sol:570:      for (uint256 i = 0; i < users.length; ++i) {  
.solidity/Bucket.sol:27:      for (uint16 i = 0; i < stock.length; ++i) {
```



## BKTP04

类型	TODOs
严重程度	■ 低
文件	Avatar.sol
实例	1
状况	已解决

### 描述

代码中留有TODO。虽然这不会造成任何直接的问题，但它表明了一种不好的气味 和不确定性，使审计师更难对代码库做出假设。

### 补救措施

删除TODO

### 快照



A screenshot of a terminal window showing a Solidity file named Avatar.sol. The code contains a TODO comment at line 747: `// how many days passed #TODO: need to confirm the logic`. The terminal also shows three colored dots (red, yellow, green) indicating the status of the build or test process.



## 附录

### 采用的审计方法和手段

区块审计报告团队对该项目进行了严格的测试，包括对代码设计模式的分析，我们审查了智能合约架构，以确保其结构与安全使用标准的继承合约和库。我们的团队还对智能合约进行了正式的逐行检查，即人工审查，以发现潜在的问题，包括但不限于

- 竞赛条件
- 零竞赛条件批准攻击
- 重入性
- 事务排序依赖性
- 时间戳依赖性
- 检查-效果-互动模式（乐观的核算）
- 去中心化的拒绝服务攻击
- 安全乙醚传输模式
- 守护检查模式
- 故障安全模式
- 气体限值和无限循环
- 呼叫堆栈深度

在单元测试阶段，我们针对合同中的每个功能编码/进行自定义单元测试，以验证客户声称的功能。在自动测试中，我们用我们的标准多功能工具集测试智能合同，以确定漏洞和安全缺陷。



## 问题类别。

本报告中的每个问题都被分配了一个严重性等级，具体如下。

### 关键的严重性问题

关键严重性问题使智能合约容易受到重大漏洞的攻击，并可能导致资产损失和数据丢失。这些会对智能合约的功能/性能产生重大影响。

我们建议在进入MainNet之前必须修复这些问题。

### 高严重性问题

高严重度的问题不容易被利用，但它们可能会危及智能合约的执行，并有可能造成关键问题。

强烈建议在进行MainNet之前修复这些问题。

### 中等严重性问题

这个级别的问题不是导致智能合约漏洞的主要原因，它们不能导致数据操纵或资产损失，但可能影响功能。

在进入MainNet之前，解决这些问题很重要。

### 低严重性的问题

这个级别的问题对智能合约的整体功能和执行的影响非常小。这些主要是代码级别的违规行为或不恰当的格式化。

这些问题可以保持不被修复，或者在以后代码被重新部署或分叉时可以被修复。

### 信息化的结果

这些是我们的团队在手动审查智能合约时遇到的发现，这些发现对于合约的所有者和使用者来说都是很重要的。

在我们发布报告之前，这些问题必须得到所有者的认可。

### 所有权特权

智能合约的所有者在部署智能合约时可以包括某些权利和特权，这些权利和特权可能隐藏在代码库的深处，并可能使项目容易受到地毯式的攻击或其他类型的骗局。

我们BlockAudit相信透明度，因此我们分别展示了所有权的特权，以便所有者和投资者可以更好地了解项目的情况。



## 免责声明

这是一份有限的报告，是根据我们的分析结果，按照截至本报告日期的良好行业惯例，对基于智能合约的框架和算法中的网络安全漏洞和问题进行的分析，其细节在本报告中列出。为了全面了解我们的分析，客户阅读报告全文是至关重要的。虽然我们在进行分析和制作本报告时已经尽了最大努力，但需要注意的是，客户不应依赖本报告，也不能根据本报告所说的或没有所说的，或我们如何制作本报告而对我们提出索赔，客户在做出任何决定之前，必须进行自己的独立调查。我们在下面的免责声明中对此进行了详细说明，请务必全文阅读。

通过阅读本报告或其任何部分，客户同意本免责声明的条款。如果客户不同意这些条款，那么请立即停止阅读本报告，并删除和销毁客户下载和/或打印的本报告的任何/所有副本。本报告只提供信息，并停留在不依赖的基础上，不构成投资建议。任何人/没有任何权利依赖本报告或其内容，BlockAudit及其附属机构（包括控股公司、股东、子公司、雇员、董事、官员和其他代表）。

(BlockAudit)对客户或任何其他人不承担任何责任，BlockAudit也不对报告的准确性或完整性向任何人提出任何保证或陈述。本报告 "按原样 "提供，除本免责声明所述外，没有任何条件、保证或其他任何形式的条款，BlockAudit在此排除所有陈述、保证、条件和其他条款（包括但不限于法律所隐含的质量令人满意、适用于目的和使用合理谨慎和技能的保证），如果没有本条款，这些保证可能对报告产生影响。



除法律禁止的情况外，BlockAudit在此排除所有的责任和义务，客户或任何其他人都不得对BlockAudit提出任何数额或种类的损失或损害的索赔，这些损失或损害可能导致客户或任何其他人（包括但不限于任何直接、间接、特殊、惩罚性、后果性或纯粹的经济损失或损害，或任何收入、利润、数据、合同的损失。商誉、数据、合同、资金使用或业务中断的损失，以及无论在任何司法管辖区内因本报告和使用、无法使用本报告或使用本报告的结果以及对本报告的任何依赖而以任何方式引起或与之相关的侵权行为（包括但不限于疏忽）、合同、违反法定义务、失实陈述（无论是无辜还是疏忽）或其他任何性质的索赔。

对证券的分析纯粹是基于收到的智能合约。没有相关/第三方智能合约、应用程序或操作的安全性进行了审查。没有审查过任何产品代码。

注意：本文件中的陈述不应被解释为投资或法律建议，其作者也不应对根据这些陈述做出的决定负责。确保智能合约的安全是一个多步骤的过程。不能认为一次审计就足够了。我们建议AVATAR团队制定一个错误赏金计划，以鼓励其他第三方对智能合约进行进一步分析



## 关于BlockAudit

BlockAudit是一个行业领先的安全组织，帮助基于Web3区块链的项目提高其智能合约的安全性和正确性。凭借多年的经验，我们有一个专门的团队，能够用各种语言进行审计，包括HTML、PHP、JS、Node、React、Native、Solidity、Rust和其他Web3框架，用于DApps、DeFi、GameFi和Metaverse平台。

BlockAudit的使命是使Web3成为一个安全的地方，它致力于为合作伙伴提供预算和投资者友好的安全审计报告，这将大大增加其项目的价值。



[www.blockaudit.report](http://www.blockaudit.report)



[team@blockaudit.report](mailto:team@blockaudit.report)



[@BlockAudit](https://twitter.com/BlockAudit)



[github.com/Block-Audit-Report](https://github.com/Block-Audit-Report)

