Web3 项目安全手册

Web3 Project Security Handbook

Web3 项目安全手册

Web3 Project Security Handbook

1

Web3 项目安全实践要求

Web3 Project Security Practice Requirements

2

慢雾智能合约审计技能树

Learning Roadmap for Becoming a Smart Contract Auditor

3

基于区块链的加密货币安全审计指南

Blockchain-Based Cryptocurrency Security Audit Guide

4

加密资产安全解决方案

Cryptocurrency Security Solution by SlowMist

1

Web3 项目安全实践要求

Web3 Project Security Practice Requirements

v0.1

T41nk@SlowMist Team

SLOWMIST

https://github.com/slowmist/Web3-Project-Security-Practice-Requirements

Web3 项目安全实践要求

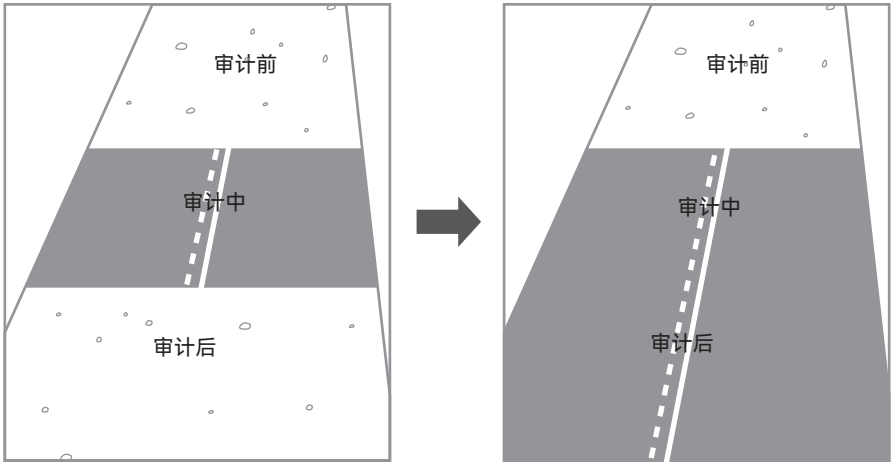
中文版

0x00 背景概述

现今针对 Web3 项目的攻击手法层出不穷，且项目之间的交互也越发复杂，在各个项目之间的交互经常会引入新的安全问题，而大部分 Web3 项目研发团队普遍缺少的一线的安全攻防经验，并且在进行 Web3 项目研发的时候重点关注的是项目整体的商业论证以及业务功能的实现，而没有更多的精力完成安全体系的建设，因此在缺失安全体系的情况下很难保证 Web3 项目在整个生命周期的安全性。

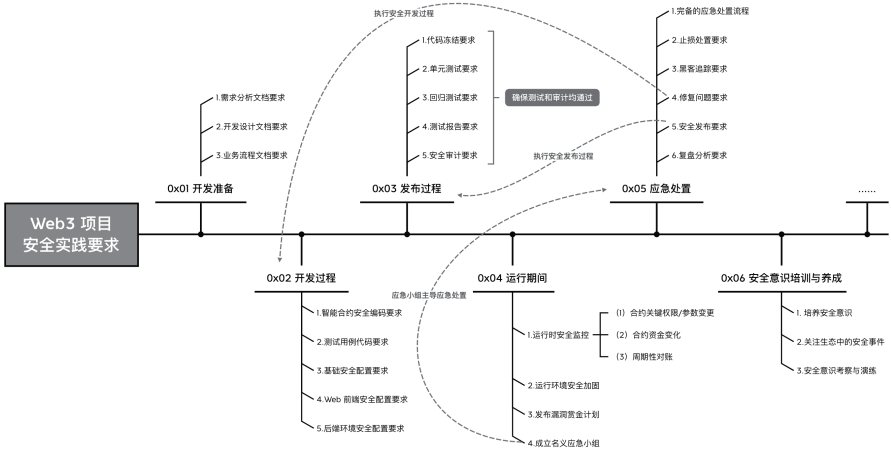
通常项目方团队为了确保 Web3 项目的安全会聘请优秀的区块链安全团队对其代码进行安全审计，在进行安全审计的时候，才能够更好地实现各种安全实践要求，但是区块链安全团队的审计仅仅是短期的引导，并不能让项目方团队建立属于自己的安全体系。

因此慢雾安全团队开源了 Web3 项目安全实践要求来持续性帮助区块链生态中的项目方团队掌握相应的 Web3 项目的安全技能，希望项目方团队能够基于 Web3 项目安全实践要求建立和完善属于自己的安全体系，在审计之后也能具备一定的安全能力。



Web3 项目安全实践要求将包含如下的内容，目前属于 v0.1 版本，并且还在持续的完善，如果你有更好的建议欢迎提交反馈。

如果你需要帮助请联系 **team@slowmist.com**，**sec_audit@slowmist.com**



0x01 开发准备

1. 需求分析文档要求

- 确保包含项目的详尽描述;
- 确保包含项目解决的问题;
- 确保包含安全/隐私风险评估。

2. 开发设计文档要求

- 确保包含项目的架构设计图;
- 确保包含代码中函数的功能描述;
- 确保包含代码中合约之间的关联关系描述;
- 确保安全/隐私的要求被正确实施。

3. 业务流程文档要求

- 确保包含项目中每个业务流程的描述;
- 确保包含详尽的业务流程图;
- 确保包含详尽的资金链路图。

0x02 开发过程

1. 智能合约安全编码要求

- 确保包含尽可能基于 OpenZeppelin 等知名 library 进行开发;
- 确保包含使用 SafeMath 或 0.8.x 以上的编译器版本来避免绝大部分溢出问题;
- 确保遵循函数命名规范，参考: **Solidity Style Guide *** ;
- 确保函数和变量可见性采用显性声明;

- 确保函数返回值被显性赋值；
- 确保函数功能和参数注释完备；
- 确保外部调用正确检查返回值，包含：transfer，transferFrom，send，call，delegate-call 等；
- 确保 interface 的参数类型返回值等实现是正确的；
- 确保设置合约关键参数时有进行鉴权并使用事件进行记录；
- 确保可升级模型的新的实现合约的数据结构与旧的实现合约的数据结构是兼容的；
- 确保代码中涉及算数运算的逻辑充分考虑到精度问题，避免先除后乘导致可能的精度丢失的问题；
- 确保 call 等 low level 调用的目标地址和函数是预期内的；
- 使用 call 等 low level 调用的时候要根据业务需要限制 Gas；
- 编码规范进行约束，遵循：先判断，后写入变量，再进行外部调用 (Checks-Effects-Interactions)；
- 确保业务上交互的外部合约是互相兼容的，如：通缩/通胀型代币，ERC-777，ERC-677，ERC-721 等可重入的代币，参考: **重入漏洞案例 *** ；
- 确保外部调用充分考虑了重入的风险；
- 避免使用大量循环对合约的 storage 变量进行赋值/读取；
- 尽可能避免权限过度集中的问题，特别是修改合约关键参数部分的权限，要做权限分离，并尽可能采用治理，timelock 合约或多签合约进行管理；
- 合约的继承关系要保持线性继承，并确保继承的合约业务上确实需要；
- 避免使用链上的区块数据作为随机数的种子来源；
- 确保随机数的获取和使用充分考虑回滚攻击的可能；
- 尽量使用 Chainlink 的 VRF 来获取可靠的随机数，参考: **Chainlink VRF *** ；
- 避免使用第三方合约的 token 数量直接计算 LP Token 价格，参考: **如何正确获取 LP 的价 *** ；
- 通过第三方合约获取价格的时候避免单一的价格来源，建议采用至少 3 个价格来源；
- 尽可能在关键的业务流程中使用事件记录执行的状态用于对项目运行时的数据分析；
- 预留全局与核心业务紧急暂停的开关，便于发生黑天鹅事件的时候及时止损。

2. 测试用例代码要求

- 确保包含业务流程/函数功能可用性测试；
- 确保包含单元测试覆盖率 95% 以上，核心代码覆盖率要达到 100% 。

3. 基础安全配置要求

- 确保使用有效的 CI/CD Pipeline 参考：[CI/CD Pipeline for Smart Contracts](https://docs.tenderly.co/forks/guides/ci-cd-pipeline-for-smart-contracts)；
- 确保官方邮箱使用知名服务商，如：Gmail；
- 确保官方邮箱账号强制开启 MFA 功能；
- 确保使用知名域名服务商，如：GoDaddy；
- 确保域名服务商平台的账号开启 MFA 安全配置；
- 确保使用优秀的 CDN 服务提供商，如：Akamai、Cloudflare 等；
- 确保 DNS 配置开启了 DNSSec，在域名服务管理平台上为管理账号设置强口令并开启 MFA 认证；
- 确保 DNS 解析使用优秀的域名服务商，如：GoDaddy、NameSilo、NameCheap 等；
- 确保开启域名隐私保护；
- 确保全员的手机和电脑设备使用杀毒软件，如：卡巴斯基、AVG 等。

4. Web 前端安全配置要求

- 确保全站的 HTTP 通讯采用 HTTPS；
- 确保配置了HSTS，以防止中间人攻击，如：DNS hijacking，BGP hijacking，参考：**HSTS 配置介绍 *** ；
- 确保配置了 X-FRAME-OPTIONS，以防止 Clickjacking 攻击，参考：**X-FRAME-OPTIONS 配置介绍 *** ；

- 确保配置了 X-Content-Type-Options，以对抗浏览器 sniff 行为导致的风险，参考：**X-Content-Type-Options 配置介绍 ***；
- 确保配置了 CSP 策略，以防止 XSS 攻击，参考：**CSP 内容安全策略介绍 ***；
- 确保与权限和用户凭证相关的 Cookie 配置了 HttpOnly， Secure，Expires，SameSite 标志，参考：**Cookie 配置介绍 ***；
- 确保不同业务的子域严格划分开，避免子域的 XSS 问题互相影响；
- 确保引用的第三方资源使用了 integrity 属性进行限制，避免第三方被黑导致项目方的站点受到影响，参考：**SRI 配置介绍 ***；
- 确保正确配置 CORS，仅允许指定 origin 域，协议和端口访问项目的资源，参考：**CORS 配置介绍 ***；
- 确保业务中实现的 addEventListener/postMessage 有检查消息的 origin 和 target，参考：**postMessage 安全介绍 ***。

5. 后端环境安全配置要求

- 确保选用优秀的云服务器提供商，如：AWS、Google 云等；
- 确保项目使用到的云平台管理账号使用强口令并开启 MFA 认证；
- 确保项目代码部署到服务器前对服务器进行安全加固，如：安装 HIDS，采用 SSH Key 进行登录，设置 SSH 登录 alert，设置 SSH 登录 google-auth 等；
- 确保使用专业软件监控服务、服务器可用性，如：APM、Zabbix；
- 确保使用专业的机构定期测试项目安全性，如：SlowMist、Trail of Bits 等。
- 确保开启服务器日志，Web 访问日志，数据库操作日志，中间件日志，并统一采集和管理可以使用 splunk (https://www.splunk.com/) 的解决方案；
- 确保项目的服务器做好网络的访问限制，通过 IP 白名单地址和网段划分，对项目的网络架构进行优化和规范，仅允许业务上需要的服务器之间的网络互通。

0x03 发布过程

需要有完备的安全上线发布流程，可以参考如下的内容进行细化。

1. 代码冻结要求

在预计的上线时间倒推 2 天，即上线 2 天前必须冻结代码不再做任何代码改动。

2. 单元测试要求

确保单元测试覆盖率 95% 以上，核心代码覆盖率 100%;

确保输出单元测试的覆盖率报告。

3. 回归测试要求

在上线 1 天前执行单元测试并进行回归测试。

4. 测试报告要求

上线前 0.5 天由开发及测试共同完成测试报告，如果不通过（含单元测试、回归测试），则推迟上线时间，开发完成修改后重新进入代码冻结阶段（即推迟至少 2 天）。

5. 安全审计要求

- 培养内部安全团队，每次新的代码都需要经过内部安全团队进行 code review，同时不断提高和培养内部安全团队的能力，参考：[《SlowMist-Learning-Roadmap-for-Becoming-a-Smart-Contract-Auditor》](https://github.com/slowmist/SlowMist-Learning-Roadmap-for-Becoming-a-Smart-Contract-Auditor);
- 安全审计人员在代码冻结后进入整体安全回归，如发现任一漏洞或安全隐患（严重、高危、中危），则推迟上线时间，开发完成修改后重新进入代码冻结（即推迟至少 2 天）；
- 安全审计需要至少三个团队进行独立的审计，可以采用 1 个内部团队 + 2 个外部团队。

0x04 运行期间

1. 运行时安全监控

- 确保项目使用到的 Netlify or Vercel 开启了 audit log or Monitoring；
- 确保设置了 DNS 变更监控，可以使用商业解决方案，如：**Better Stack *** 。

尽可能的通过关键业务流程中触发的事件来发现项目运行时的安全问题，如：

- 合约关键权限/参数变更：监控管理角色发生变更的事件，管理角色修改合约关键参数的事件，及时发现私钥可能被盗的情况；
- 合约资金变化：监控价格变动及合约资金变动的情况，及时发现可能的闪电贷等攻击；
- 周期性对账：周期性对链上的事件与交易进行对账，及时发现可能的业务逻辑上的问题。

2. 运行环境安全加固

- 确保实施前端代码所在服务器的安全加固，如：**安装 HIDS ***，采用 SSH Key 进行登录，**设置 SSH 登录 alert ***，**设置 SSH 登录 google-auth *** 等；
- 确保 DNS 配置开启了 DNS Sec，在域名服务管理平台上为管理账号设置强口令并开启 2 次认证；
- 确保项目使用到的云平台管理账号使用了强口令并开启了 2 次认证；
- 确保更好 DNS 解析等操作开了了 2 次认证。

3. 发布漏洞赏金计划

发布漏洞赏金计划或入驻知名的漏洞赏金平台，吸引社区白帽子为项目保驾护航；可以选择 **BugRap ***，**Code4rena ***，**Immunefi *** 。

4. 成立名义应急小组

成立名义应急小组并对外提供联系方式，由应急小组负责处理白帽子发现的问题或在黑天鹅事件爆发时主导团队成员进行应急处置。

0x05 应急处置

1. 完备的应急处置流程

- 尽可能地制定完备地应急处置流程，有条不紊地根据应急处置流程来处置黑天鹅事件；
- 周期性进行安全应急响应演练，改进流程以确保正确且快速地响应安全事件，如：SEAL Drills * 。

2. 止损处置要求

- 根据问题影响的范围和危害程度，及时通过紧急暂停开关进行止损；
- 通知社区成员发生黑天鹅事件，避免用户继续与项目进行交互导致亏损。

3. 黑客追踪要求

- 迅速分析黑客的获利地址，并留存 PC / Web / 服务器的访问日志（如果有木马请留存木马文件）；
- 对服务器进行快照，及时保留被黑现场；
- 联系专业的安全团队协助进行追踪，如：**MistTrack 追踪分析平台 *** ，**Chainalysis *** 。

4. 修复问题要求

- 与专业安全团队讨论问题的最佳修复方案；
- 正确实施修复方案并请专业的安全团队进行验证。

5. 安全发布要求

执行发布过程要求，确保一切代码的变更均有经过测试和安全审计。

6. 复盘分析要求

- 披露验尸报告并与社区成员同步修复方案及补救措施；
- 验尸报告需要同步问题的本质原因，问题的影响范围，具体的损失，问题的修复情况，黑客的追踪等相关进展。

0x06 安全意识培训与养成

1. 培养安全意识

- 团队成员要仔细详尽阅读**《区块链黑暗森林自救手册》*** 培养相关的安全意识；
- 可以通过一些在线的安全意识测试网站来进行培训和测验，如：**Google's Phishing Quiz***，**Phishing.org***；
- 同时在团队管理中应当加入安全意识培训和考核的环节，如：新成员入职要进行安全意识的培训和测验，如果是IT岗位的人员还应当对代码开发，系统运维等维度的安全意识进行培训和考核；
- 并且周期性地组织安全意识培训，将近期黑客新的攻击手法和相关的安全事件进行同步。

2. 关注生态中的安全事件

- 关注 **ScamSniffer***，**Wallet Guard*** 这类专注于 Crypto Anti-Scam 团队的动态，了解当下流行的钓鱼攻击手法；

- 关注社区或者生态中的安全事件，整理并同步给团队成员。

3. 安全意识考察与演练

- 周期性对团队成员的安全意识进行考察，可以和优秀的安全公司合作，进行相关的安全演练；
- 可以通过模拟黑客的攻击方式对团队成员进行钓鱼或投放木马来对团队成员的安全意识进行考察，同时也能够对团队成员设备上的终端安全防护系统进行检测能力的验证。

*

Solidity Style Guide	https://docs.soliditylang.org/en/v0.8.14/style-guide.html
重入漏洞案例	https://medium.com/amber-group/preventing-re-entrancy-attacks-lessons-from-history-c2d96480fac3
Chainlink VRF	https://docs.chain.link/docs/chainlink-vrf/
如何正确获取 LP 的价	https://blog.alphafinance.io/fair-lp-token-pricing/
HSTS 配置介绍	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security
X-FRAME-OPTIONS 配置介绍	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
X-Content-Type-Options 配置介绍	https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options
CSP 内容安全策略介绍	https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
Cookie 配置介绍	https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies
SRI 配置介绍	https://developer.mozilla.org/zh-CN/docs/Web/Security/Subresource_Integrity
CORS 配置介绍	https://developer.mozilla.org/zh-CN/docs/Web/HTTP/CORS
postMessage 安全介绍	https://developer.mozilla.org/zh-CN/docs/Web/API/Window/postMessage
Better Stack	https://betterstack.com/docs/uptime/uptime-monitor/
安装 HIDS	https://www.aliyun.com/product/aegis
设置 SSH 登录 alert	https://medium.com/@alessandrocuda/ssh-login-alerts-with-sendmail-and-pam-3ef53aca1381
设置 SSH 登录 google-auth	https://goteleport.com/blog/ssh-2fa-tutorial/
BugRap	https://bugrap.io/
Code4rena	https://code4rena.com/
Immunefi	https://immunefi.com/
SEAL Drills	https://twitter.com/samczsun/status/1717243519243636755
MistTrack 追踪分析平台	https://misttrack.io/
Chainalysis	https://www.chainalysis.com/
《区块链黑暗森林自救手册》	https://darkhandbook.io/
Google's Phishing Quiz	https://phishingquiz.withgoogle.com/
Phishing.org	https://www.phishing.org/phishing-resources
ScamSniffer	https://x.com/realScamSniffer
Wallet Guard	https://x.com/wallet_guard

Web3 Project Security Practice Requirements

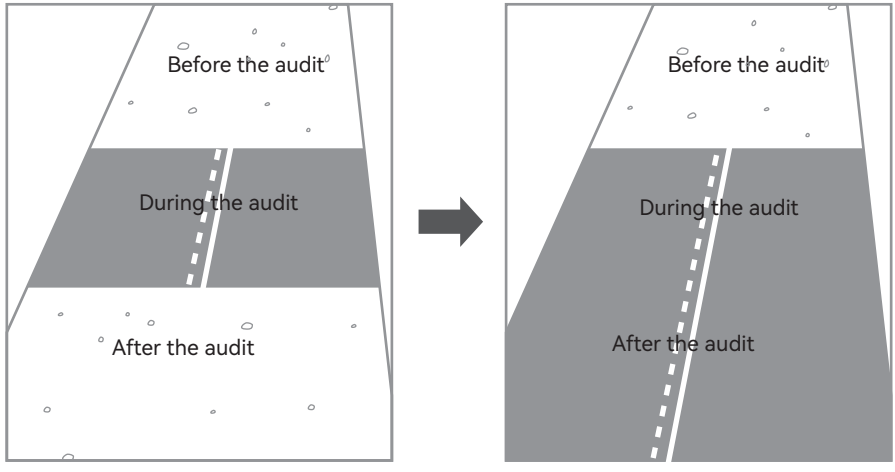
English Version

0x00 Background Overview

At present, attacks against Web3 projects emerge in an endless stream, and the interactions between projects are becoming more and more complex. The interaction between various projects often introduces new security issues, and most Web3 development teams generally lack experience in cutting-edge security attack and defense.while in the development of Web3 projects, the focus is on the business demonstration of the project and the realization of business functions, and there is no more energy to complete the construction of the security system. Therefore, in the absence of a security system, it is difficult to ensure the security of a Web3 project throughout its life cycle.

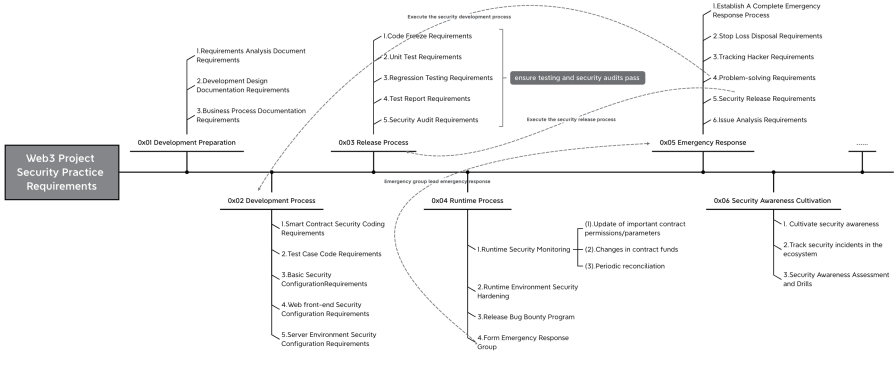
Usually, the project team will find an excellent blockchain security team to conduct security audits on its code in order to ensure the security of Web3 projects. When conducting security audits, various security practice requirements can be better achieved, but blockchain security the team's audit is only a short-term guidance, and does not allow the project team to establish its own security system.

Therefore, the SlowMist security team has open-sourced Web3 Project Security Practice Requirements to continuously help the project team in the blockchain ecosystem to master the corresponding Web3 project security skills, It is hoped that the project team can establish and improve its own security system based on Web3 Project Security Practice Requirements, and also have certain security capabilities after security audits.



Web3 Project Security Practice Requirements contains the following content, which is currently in the v0.1 version and is still being improved. If you have better suggestions, please submit feedback.

If you need help please contact **team@slowmist.com**, **sec_audit@slowmist.com**



0x01 Development Preparation

1. Requirements Analysis Document Requirements

- Make sure to include a thorough description of the project;
- Make sure to include the specific problem that the project is to address;
- Make sure to include a security/privacy risk assessment.

2. Development Design Documentation Requirements

- Make sure to include the project's architectural design diagram;
- Make sure to include functional descriptions of functions in the code;
- Make sure to include a description of the relationship between the contracts in the code;
- Make sure to include security/privacy requirements.

3. Business Process Documentation Requirements

- Make sure to include a description of each business process in the project;
- Make sure to include a detailed business process diagram;
- Make sure to include a detailed funding link diagram.

0x02 Development Process

1. Smart Contract Security Coding Requirements

- Make sure to develop based on well-known libraries such as: OpenZeppelin as much as possible;
- Make sure to include a compiler that uses SafeMath or a compiler version above 0.8.x to prevent most overflow issues;
- Make sure to follow function naming conventions, see: **Solidity Style Guide *** ;
- Make sure that function and variable visibility are explicitly declared;
- Make sure that the function return value is explicitly assigned;
- Make sure that functions and parameters are well-annotated;
- Make sure that external calls correctly check the return value, including: transfer, transferFrom, send, call, delegatecall, etc.;

- Make sure that the implementation of the parameter type and return value of the interface is correct;
- Make sure that the key parameters of the contract are set up with authentication and use events to record;
- Make sure that the data structure of the new implementation contract of the upgradeable model is compatible with the data structure of the old implementation contract;
- Make sure that the logic involved in arithmetic operations in the code fully considers the precision problem, and avoids the problem of possible loss of precision caused by dividing and then multiplying;
- Make sure that the target address and function of low-level calls such as: call are expected;
- When using low-level calls such as: call, limit Gas according to business needs;
- Coding specifications are constrained, follow: first judge, then write variables, and then make external calls (Checks-Effects-Interactions);
- Make sure that external contracts that interact in business are compatible with each other, such as: deflation/inflation tokens, reentrant tokens such as: ERC-777, ERC-677, ERC-721, see: **Reentrancy Vulnerability Case *** ;
- Make sure that external calls fully consider the risk of reentrancy;
- Avoid using a lot of loops to assign/read the contract's storage variable;
- Avoid the problem of excessive concentration of authority as much as possible, especially the authority to modify the key parameters of the contract, separate authority, and use governance, timelock contract or multi-signature contract to manage as much as possible;
- The inheritance relationship of contracts should maintain linear inheritance, and ensure that the inherited contracts are really needed for business;
- Avoid using on-chain block data as a seed source for random;
- Make sure that the acquisition and use of random numbers fully consider the possibility of rollback attacks;
- Use Chainlink's VRF to obtain reliable random, see: **Chainlink VRF *** ;
- Avoid using the token quantity of the third-party contract to directly calculate the LP Token price, see: **How to get the price of LP correctly *** ;
- Avoid a single price source when obtaining prices through third-party contracts. It is recommended to use at least 3 price sources;
- Use events as far as possible in key business processes to record the status of execution for data analysis when the project is running;
- Reserve the switch for an emergency suspension of the global and core business, so that it is convenient to stop losses in time when a black swan event occurs.

2. Test Case Code Requirements

- Make sure to include business process/function functional usability testing;
- Make sure that the unit test coverage rate is more than 95%, and the core code coverage rate must reach 100%.

3. Basic Security Configuration Requirements

- Make sure that use of an effective CI/CD Pipeline, see: [CI/CD Pipeline for Smart Contracts](https://docs.tenderly.co/forks/guides/ci-cd-pipeline-for-smart-contracts);
- Make sure that the official email uses well-known service providers, such as: Gmail;
- Make sure that the official email account opens MFA function;
- Make sure that the use of well-known domain name service providers, such as: GoDaddy;
- Make sure that the use of excellent CDN service providers, such as: Akamai and Cloudflare;
- Make sure that DNS configuration turns on DNSSec, set a strong password for the management account on the domain name service management platform, and turn on MFA authentication;
- Make sure that DNS resolution uses excellent domain name service providers, such as: GoDaddy, NameSilo, NameCheap, etc.;
- Make sure that domain privacy protection is turned on;
- Make sure that all mobile phones and computer devices use anti-virus software, such as: Kaspersky, AVG, etc.

4. Web front-end Security Configuration Requirementst

- Make sure that the HTTP communication of the whole site adopts HTTPS;
- Make sure HSTS is configured to prevent man-in-the-middle attacks like: DNS hijacking, BGP hijacking,see：**HSTS Configuration Introduction *** ;
- Make sure X-FRAME-OPTIONS is configured to prevent Clickjacking attacks, see: **X-FRAME-OPTIONS Configuration Introduction *** ;
- Make sure X-Content-Type-Options are configured to combat the risks caused by browser sniff behavior, see：**X-Content-Type-Options Configuration Introduction *** ;
- Make sure CSP policies are configured to prevent XSS attacks, see: **CSP Configuration Introduction *** ;

- Make sure cookies related to permissions and user credentials are configured with HttpOnly, Secure, Expires, SameSite flags, see: **Cookie Configuration Introduction *** ;
- Make sure that the sub-domains of different businesses are strictly separated to avoid the XSS problems of the sub-domains affecting each other;
- Make sure that the referenced third-party resources are restricted by the integrity attribute, so as to avoid the third-party being hacked and the project's site from being affected, see: **SRI Configuration Introduction *** ;
- Make sure CORS is properly configured to only allow the specified origin domain, protocol and port to access the project's resources, see: **CORS Configuration Introduction *** ;
- Make sure that the addEventListener/postMessage implemented in the business has the origin and target of the check message, see: **postMessage Configuration Introduction ***.

5. Server Environment Security Configuration Requirements

- Make sure that the selection of excellent cloud server providers, such as: AWS, Google Cloud, etc.;
- Make sure that the cloud platform management account used by the project uses a strong password and turns on MFA;
- Make sure that the security reinforcement of the server before the project code is deployed to the server, such as: installing HIDS, using SSH Key to log in, setting SSH login alert, setting SSH login google-auth, etc.;
- Make sure that the use of professional software monitoring services and server availability, such as: APM and Zabbix;
- Make sure that the use of professional institutions to regularly test the safety of projects, such as: SlowMist, Trail of Bits, etc;
- Make sure that server logs, web access logs, database operation logs, and middleware logs are enabled. Utilize a solution like splunk (https://www.splunk.com/) for centralized log collection and management;
- Make sure that project servers have network access restrictions in place. Utilize IP whitelists and network segmentation to optimize and regulate the project's network architecture, ensuring network connectivity only between servers required for business operations.

0x03 Release Process

A complete security online release process is required, which can be refined by referring to the following content.

1. Code Freeze Requirements

The estimated launch time is pushed back 2 days, that is, the code must be frozen and no code changes will be made 2 days before the launch.

2. Unit Test Requirements

- Make sure that the unit test coverage rate is above 95%, and the core code coverage rate is 100%;
- Make sure to output coverage reports for unit tests.

3. Regression Testing Requirements

Make sure to output coverage reports for unit tests.

4. Test Report Requirements

0.5 days before the launch, the development and testing will jointly complete the test report. If the test report is not passed (including unit testing and regression testing), the launch time will be postponed, and the code freezing stage will be re-entered after the development is completed and modified (that is, postponed for at least 2 days).

5. Security Audit Requirements

- Create an internal security team. All new code implementations must be code-reviewed by the internal security team. Continuously refine and cultivate the internal security team's competencies, see: SlowMist-Learning-Roadmap-for-Becoming-a-Smart-Contract-Auditor;
- Security auditors enter the overall security regression after the code is frozen. If any vulnerability or security risk (serious, high-risk, medium-risk) is found, the launch time will be postponed, and the code will be frozen again after the development and modification are completed (that is, delayed for at least 2 days);

- Security audit requires at least three teams to conduct independent audits, and can use 1 internal team + 2 external teams.

0x04 Runtime Process

1. Runtime Security Monitoring

- Make sure that the Netlify or Vercel used by the project has audit log or Monitoring turned on;
- Make sure that DNS change monitoring is set up and commercial solutions can be used, such as: **Better Stack *** .

As far as possible through the events triggered in the key business processes to discover the security problems of the project runtime, such as:

- Update of important contract permissions/parameters: monitor the events that the management role changes, and the events that the management role modifies the key parameters of the contract, and promptly discover the possible theft of the private key;
- Changes in contract funds: monitor price changes and changes in contract funds, and timely detect possible attacks such as: flash loans;
- Periodic reconciliation: Periodically reconcile events and transactions on the chain to discover possible business logic problems in a timely manner.

2. Runtime environment security hardening

- Make sure the security hardening of the server where the front-end code is located,such as: **Install HIDS *** , Login with SSH Key, **Set SSH login alert *** , **Set up SSH login to google-auth *** etc.;
- Make sure that DNS Sec is enabled in the DNS configuration, set a strong password for the management account on the domain name service management platform and enable 2 authentications;
- Make sure that the cloud platform management account used by the project uses a strong password and has 2 authentications enabled;
- Make sure that two-factor authentication is enabled for improved DNS resolution and other operations.

3. Release Bug Bounty Program

Publish a bug bounty program or enter a well-known bug bounty platform to attract community white hats to escort the project; you can choose: **BugRap *** , **Code4rena *** , **Immunefi *** .

4. Form Emergency Response Group

Set up a nominal emergency response group and provide contact information to the outside world. The emergency response group is responsible for dealing with problems found by white hats or leading team members to carry out an emergency response when a black swan event occurs.

0x05 Emergency Response

1. Establish A Complete Emergency Response Process

- Develop a complete emergency response process as much as possible, and deal with black swan events in an orderly manner according to the emergency response process;
- Regularly conduct security emergency response drills, refining procedures to ensure a prompt and accurate response to security incidents, such as: **SEAL Drills *** .

2. Stop Loss Disposal Requirements

- According to the scope of the problem and the degree of harm, stop the loss through the emergency pause switch in time;
- Notify community members of black swan events to prevent users from continuing to interact with the project and cause losses.

3. Tracking Hacker Requirements

- Quickly analyze the hacker's profitable address, and keep the access log of the PC/Web/-server (if there is a Trojan, please keep the Trojan file);
- Take a snapshot of the server and keep the hacked scene in time;
- Contact a professional security team to assist in tracking, such as: **MistTrack *** , **Chainalysis *** .

4. Problem-solving Requirements

- Discuss the best fixes for the problem with a professional security team;
- Correctly implement the fixed plan and ask a professional security team to verify it;

5. Security Release Requirements

Enforce release process requirements to ensure all code changes are tested and security audited.

6.Issue Analysis Requirements

- Disclose autopsy reports and synchronize restoration plans and remedies with community members;
- The autopsy report needs to synchronize the essential cause of the problem, the scope of the problem, the specific loss, the repair of the problem, the tracking of the hacker and other related progress.

0x06 Security Awareness Cultivation

1. Cultivate security awareness

- Project team members must thoroughly review the **Blockchain Dark Forest Selfguard Handbook *** to cultivate a strong security awareness.
- Online security awareness training and testing can be conducted through various websites, such as **Google's Phishing Quiz *** and **Phishing.org ***;
- Security awareness training and assessment should be integrated into team management practices. For instance, new hires should undergo security awareness training and testing, and IT personnel should receive additional training and assessment in areas such as code development and system operation security;
- Regularly conduct security awareness training to keep team members informed about the latest hacker attack techniques and related security incidents;

2. Track security incidents in the ecosystem

- Follow the updates from Crypto Anti-Scam teams like **ScamSniffer *** and **Wallet Guard *** to stay informed about the latest phishing attack techniques;
- Stay informed about security incidents within the community or ecosystem, compile information, and share it with team members;

3. Security Awareness Assessment and Drills

- Conduct periodic security awareness assessments for team members. Collaborate with reputable security companies to conduct relevant security drills;
- Security awareness assessments for team members can be conducted by simulating hacker attack methods, such as phishing or Trojan horse attacks. This not only evaluates team members' security awareness but also verifies the detection capabilities of endpoint security protection systems on their devices;

*

Solidity Style Guide
https://docs.soliditylang.org/en/v0.8.14/style-guide.html

Reentrancy Vulnerability Case
https://medium.com/amber-group/preventing-re-entrancy-attacks-lessons-from-history-c2d96480fac3

Chainlink VRF
https://docs.chain.link/docs/chainlink-vrf/

How to get the price of LP correctly
https://blog.alphafinance.io/fair-lp-token-pricing/

HSTS Configuration Introduction
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security

X-FRAME-OPTIONS Configuration Introduction
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

X-Content-Type-Options Configuration Introduction
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options

CSP Configuration Introduction
https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP

Cookie Configuration Introduction
https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

SRI Configuration Introduction
https://developer.mozilla.org/zh-CN/docs/Web/Security/Subresource_Integrity

CORS Configuration Introduction
https://developer.mozilla.org/zh-CN/docs/Web/HTTP/CORS

postMessage Configuration Introduction
https://developer.mozilla.org/zh-CN/docs/Web/API/Window/postMessage

Better Stack
https://betterstack.com/docs/uptime/uptime-monitor/

Install HIDS
https://www.aliyun.com/product/aegis

Set SSH login alert
https://medium.com/@alessandrocuda/ssh-login-alerts-with-sendmail-and-pam-3ef53aca1381

Set up SSH login to google-auth
https://goteleport.com/blog/ssh-2fa-tutorial/

BugRap
https://bugrap.io/

Code4rena
https://code4rena.com/

Immunefi
https://immunefi.com/

SEAL Drills
https://twitter.com/samczsun/status/1717243519243636755

MistTrack
https://misttrack.io/

Chainalysis
https://www.chainalysis.com/

Blockchain Dark Forest Selfguard Handbook
https://darkhandbook.io/

Google's Phishing Quiz
https://phishingquiz.withgoogle.com/

Phishing.org
https://www.phishing.org/phishing-resources

ScamSniffer
https://x.com/realScamSniffer

Wallet Guard
https://x.com/wallet_guard

2

慢雾智能合约审计技能树

Learning Roadmap for Becoming a Smart Contract Auditor

Kong'@SlowMist Team

https://github.com/slowmist/SlowMist-Learning-Roadmap-for-Becoming-a-Smart-Contract-Auditor

慢雾智能合约审计技能树

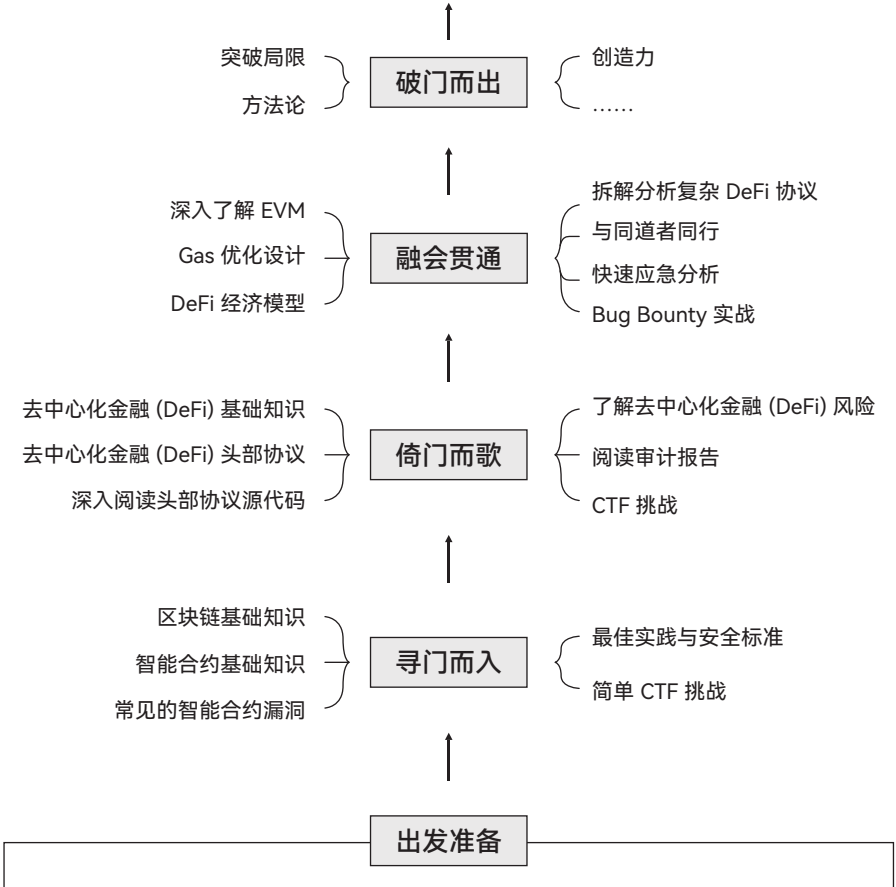
中文版

前言

本技能表是慢雾安全团队智能合约安全审计工程师的技能集合，旨在为团队成员列出智能合约安全审计的所需技能并驱动团队成员形成研究、创造、工程的自我进化思维。

智能合约安全审计技能主要分为四个部分：寻门而入、倚门而歌、融会贯通、破门而出，由浅至深地列出在各个阶段所需掌握的专业技能。而在此之前，需要一些通用技能武装我们的大脑，出发准备部分将会是我们审计之路的锚点。

路线图



知行合一

- 学习应有所输出，输出应有所实践
- 慢雾安全团队知识库
- Github 开源文化

守正出奇

- 坚守底线
- 负责任的披露
- 黑客思维

团队意识

- 协同审计流
- 审计工作流
- Hacking Time 文化

出发准备

所谓磨刀不误砍柴功，在正式出发之前强化自己的思维是必要的，这可以使我们走得坚定、走得更远。

1. 知行合一

认知与实践是密不可分的，理论与实践应相统一。

- 学习应有所输出，输出应有所实践
- 慢雾安全团队知识库（https://github.com/slowmist/Knowledge-Base）
- Github 开源文化

2. 守正出奇

道德和法律是安全从业者的底线，安全从业者在坚守底线的同时也要锻造过硬的技术，在关键时刻出奇制胜。

- **坚守底线**

  审计人员应遵守法律，坚守道德底线。

- **负责任的披露**

  慢雾 (SlowMist) 预警流程；

  FIRST 道德守则（https://www.first.org/global/sigs/ethics/ethics-first）。

- **黑客思维**

  坚守底线的同时出奇制胜；

  守正：保持敬畏，坚守底线；

  出奇：脑洞要大，心要细，反向思维，开放性思维。

3. 团队意识

单个人的能力覆盖面总是有限，团队战斗可以很好地补全个人的不足。

- **协同审计流**

  SlowMist MistPunk 审计工作台做协同，通过技术的方式保证审计质量，同时沉淀审计经验。

- **审计工作流**

  SlowMist 审计工作流程，通过管理方式保证审计质量，同时为审计工作查缺补漏。

- **Hacking Time 文化**

  团队成员随时随地的思维碰撞与分享，通过思维碰撞和分享对齐团队能力，提升团队整体能力。

寻门而入

加密世界发展至今其涵盖了密码学、经济学、数据科学等学科，面对知识体量极为庞大的加密世界，如何寻门而入是为关键。本阶段将从 Ethereum (以太坊) 及其智能合约语言 Solidity 开始寻找进入加密货币世界的大门。

1. 区块链基础知识

在了解智能合约是什么之前，应该先了解智能合约所运行的区块链平台是什么。

- 什么是区块链？（https://www.investopedia.com/terms/b/blockchain.asp）
- 区块链可视化演示（https://www.youtube.com/watch?v=_160oMzblY8）
- **慢雾 (SlowMist) 区块链入门科普 ***
- 加密货币工作原理（https://www.bilibili.com/video/BV11x411i72w/）
- 阅读《精通比特币》（https://github.com/inoutcode/bitcoin_book_2nd）
- 阅读《精通以太坊》（https://github.com/inoutcode/ethereum_book）

  当前应着重阅读第 1、4、5、6、7 和 13 章节

2. 智能合约基础知识

在不同的区块链可能会使用不同的语言构建智能合约，例如：Solidity、Move、Rust、Vyper、Cairo、C++ 等。目前 EVM 兼容链使用的 Solidity 仍是最流行且易于入门的智能合约语言，应该确保完整阅读完其语言文档。且应了解运行在 Ethereum(以太坊) 上的代币合约的设计标准与具体的合约实现。在此基础上了解智能合约是如何做到可升级的，并实操掌握智能合约的编写与测试。

- Solidity 官方文档 （https://docs.soliditylang.org/en/latest/）
- 阅读《精通以太坊》

  当前应着重阅读剩余的其他章节
- 了解基础的以太坊意见征求稿 ERC（https://eips.ethereum.org/erc）

  ERC20（https://eips.ethereum.org/EIPS/eip-20）同质化代币标准

  ERC165（https://eips.ethereum.org/EIPS/eip-165）接口标准

  ERC173（https://eips.ethereum.org/EIPS/eip-173）合约所有权标准

  ERC191（https://eips.ethereum.org/EIPS/eip-191）数据签名标准

  ERC601（https://eips.ethereum.org/EIPS/eip-601）确定性钱包分层结构标准

  ERC721（https://eips.ethereum.org/EIPS/eip-721）非同质化代币标准

  ERC777（https://eips.ethereum.org/EIPS/eip-777）可交互性代币标准

  ERC1155（https://eips.ethereum.org/EIPS/eip-1155）多代币标准

  ERC1167（https://eips.ethereum.org/EIPS/eip-1167）最小代理合约

  ERC1967（https://eips.ethereum.org/EIPS/eip-1967）代理数据存储插槽

  ERC2612（https://eips.ethereum.org/EIPS/eip-2612）代币批准签名

  ERC4626（https://eips.ethereum.org/EIPS/eip-4626）代币金库标准
- 学习 **OpenZeppelin Token *** 部分的实现
- 了解可升级合约/代理合约是什么

  不同模式的代理合约介绍 *

  代理合约深入研究（https://proxies.yacademy.dev/pages/proxies-list/）

  OpenZeppelin Proxy * 实现文档

- 学习智能合约编写
    - WTF Solidity 智能合约教程（https://www.wtf.academy/）
    - Crypto Zombies（https://cryptozombies.io/en/course/）
    - Smart Contract Engineer（https://www.smartcontract.engineer/）
    - Solidity by Example（https://solidity-by-example.org/）
- 阅读《精通以太坊智能合约开发》
- 学习使用智能相关 Build 工具
    - 流行的在线 IDE
        - Remix（https://remix.ethereum.org/）
        - ChainIDE（https://chainide.com/）
        - Tenderly Sandbox（https://sandbox.tenderly.co/）
    - 熟悉使用包管理器
        - npm（https://www.npmjs.com/）
        - yarn（https://yarnpkg.com/）
        - pnpm（https://pnpm.io/）
    - 流行的智能合约测试和调试框架
        - Foundry（https://book.getfoundry.sh/）
            - 简便的测试工具（https://book.getfoundry.sh/forge/tests）
            - 强大的 Cheatcodes（https://book.getfoundry.sh/cheatcodes/）
            - 最佳实践（https://book.getfoundry.sh/tutorials/best-practices）
        - Hardhat（https://hardhat.org/hardhat-runner/docs/getting-started#overview）
            - 善用其强大的插件（https://hardhat.org/hardhat-runner/plugins）
        - Brownie（https://eth-brownie.readthedocs.io/en/stable/）
        - Tenderly（https://tenderly.co/）
            - 简便的开发测试环境 **DevNet ***
            - 快速**交易模拟 ***
            - **可视化的交易 Debug 工具 ***

Sentio（https://app.sentio.xyz/explorer）

用代码索引做在线 Debug *

交易模拟中覆盖合约 *

与智能合约交互

了解 JSON-RPC（https://ethereum.org/en/developers/docs/apis/json-rpc/）

ethers.js（https://docs.ethers.org/）

Web3.js（https://web3js.readthedocs.io/）

Web3.py（https://web3py.readthedocs.io/）

Viem（https://viem.sh/）

3. 常见的智能合约漏洞

在学习完成智能合约基础知识后应掌握其常见的基础漏洞，并知晓漏洞原理。其中 Quillhash 整理的漏洞列表聚合了多个来源，其较为完备的展示了当前常见的智能合约漏洞类型。(但对于初学者来说建议反复阅读所有列表以加深印象)

- DASP Top 10 （https://www.dasp.co/）
- SWC（https://swcregistry.io/）智能合约弱点分类
- 智能合约安全指南（https://scsfg.io/hackers/）
- **Kaden: 智能合约攻击向量 ***
- Quillhash: Solidity 攻击向量（https://github.com/Quillhash/Solidity-Attack-Vectors）
- **RareSkills Smart Contract Security ***

4. 最佳实践与安全标准

作为审计人员，必须了解智能合约的最佳实践以及安全标准。最佳实践为审计中寻找安全问题提供参考，安全标准为审计中提出的安全问题提供依据。

- Solidity Patterns（https://fravoll.github.io/solidity-patterns/）
- Solcurity（https://github.com/transmissions11/solcurity）

- **ConsenSys 智能合约最佳实践 ***
- **Solidity 安全陷阱和最佳实践 101 ***
- **Solidity 安全陷阱和最佳实践 201 ***
- SCSVSv2（https://github.com/securing/SCSVS/tree/prerelease/SCSVSv2）
- EEA EthTrust Certification（https://entethalliance.org/specs/ethtrust-sl/）
- Foundry 测试最佳实践（https://book.getfoundry.sh/tutorials/best-practices）

5. 简单 CTF 挑战

在学习了区块链与智能合约基础知识以及常见的智能合约漏洞后，可以通过一些简单的 CTF 挑战巩固以及实践所学的知识。

- OpenZeppelin Ethernaut（https://ethernaut.openzeppelin.com/）
- Capture the Ether（https://capturetheether.com/）

倚门而歌

掌握了区块链与智能合约的基础知识后，我们便推开了 Solidity 智能合约安全审计的大门，门后的智能合约世界仍极为广阔。本阶段将从去中心化金融 (DeFi) 开始深入地探索门后更为广阔的智能合约应用。

1. 去中心化金融(DeFi)基础知识

区块链和智能合约使 DeFi 的构建成为可能，DeFi 的出现也使得 Ethereum 等公链快速发展。在做进一步探索之前，理应了解 DeFi 是什么？

- 阅读《How To DeFi: Beginner》（https://landing.coingecko.com/how-to-defi/）
- 阅读《How To DeFi: Advanced》（https://landing.coingecko.com/how-to-defi/）
- **DeFi 各类型介绍 ***

    - 了解各个类型的 DeFi 是什么
    - 了解一些基础的经济学知识与常用术语
- **基础的金融玩法 *** 介绍
- 经济模型 101（https://tokenomicsdao.xyz/tokenomics101/）

2. 去中心化金融(DeFi)头部协议

在初步了解了 DeFi 是什么后，应进一步了解它们实现了什么/是如何实现的？通过阅读当前头部去中心化金融(DeFi)协议的技术文档以初步了解头部 DeFi 协议是如何实现的。

- MakerDAO (CDP)（https://docs.makerdao.com/）
- AAVE (Lending)（https://docs.aave.com/hub/）
    - V2（https://docs.aave.com/developers/v/2.0/）
    - V3（https://docs.aave.com/developers/getting-started/readme）
- Compound (Lending)
    - V2（https://docs.compound.finance/v2/）
    - V3（https://docs.compound.finance/）
- Uniswap (DEX)
    - V2（https://docs.uniswap.org/contracts/v2/overview）
    - V3（https://docs.uniswap.org/contracts/v3/overview）
- Curve (DEX)
    - 技术文档（https://docs.curve.fi/）
    - 算法简述（https://hackmd.io/@alltold/curve-magic）
    - **Curve 牛顿迭代详解 ***
- Chainlink (Oracle)
    - 价格预言机（https://docs.chain.link/data-feeds）
    - VRF（https://docs.chain.link/vrf/v2/introduction）
- Convex Finance (Yield)
    - 协议介绍（https://docs.convexfinance.com/convexfinance/）

    - 技术文档（https://docs.convexfinance.com/convexfinanceintegration/）
- Yearn Finance (Yield Aggregator)（https://docs.yearn.fi/getting-started/intro）
- GMX (Derivatives)（https://gmxio.gitbook.io/gmx/）
- Nexus Mutual (Insurance)（https://docs.nexusmutual.io/overview/）
- OpenSea (NFT Marketplace)（https://github.com/ProjectOpenSea/seaport#seaport）
- Set Protocol (Indexes)（https://docs.tokensets.com/）
- Lido (Liquid Staking)（https://docs.lido.fi/）
- ...

3. 深入阅读头部协议源代码

当前多数 DeFi 项目都相互依赖、组合，一些头部的 DeFi 协议成了构件 DeFi 组合基石，所以掌握这些 DeFi 的实现极为重要。在先前通过协议技术文档对 DeFi 的实现进行初步了解后，再通过阅读全量源代码的方式掌握其具体的逻辑、经济模型。

4. 了解去中心化金融(DeFi) 风险

DeFi 并不局限于智能合约，前端、后端也是其重要的组成部分，绝大部分用户通过前端与 DeFi 进行交互。因此在了解了 DeFi 的运作与实现后，通过前端安全实践、后端安全配置要求与 DeFi 历史漏洞对其面临的风险进行学习与实践。

- 了解 Web 前端安全
    - 阅读 SlowMist Web 前端最佳安全实践指南
    - 更多了解《Web 前端黑客技术揭秘》
- 了解 DeFi 被黑原因
    - **SlowMist DeFi 被黑简析 ***
    - SlowMist Medium（https://slowmist.medium.com/）
    - **DeFiHackLabs ***
    - Rekt（https://rekt.news/zh/）

- Immunefi（https://medium.com/@immunefi）
- QuillAudits（https://quillaudits.medium.com/）
- BlockSec（https://blocksecteam.medium.com/）
- Neptune Mutual（https://medium.com/@neptunemutual）
- PeckShield（https://twitter.com/peckshield）
- hacxyk（https://medium.com/@hacxyk）
- TrailOfBits（https://blog.trailofbits.com/）
- Secureum（https://secureum.substack.com/）
- Openzeppelin（https://blog.openzeppelin.com/security-audits/）
- OfferCIA（https://officercia.mirror.xyz/）

5. 阅读审计报告

在进行审计时，个人的角度总是会有所遗漏，无法覆盖所有情况。因此通过阅读他人的审计报告以学习不同的漏洞发现方式和审计思考方式很重要。

- SlowMist Audit Reports（https://github.com/slowmist/Knowledge-Base）
- Solodit Aggregation（https://solodit.xyz/）
- Code4rena Audit Reports（https://code4rena.com/reports）
- Consensys Audit Reports（https://consensys.net/diligence/audits/）
- QuillAudits Audit Reports（https://github.com/Quillhash/QuillAudit_Reports）
- Spearbit Audit Reports（https://github.com/spearbit/portfolio/tree/master/pdfs）
- Sherlock Audit Reports（https://github.com/sherlock-protocol/sherlock-reports）
- ADBK Audit Reports（https://github.com/abdk-consulting/audits）
- BlockSec Audit Reports（https://github.com/blocksecteam/audit-reports）
- Certik Audit Reports（https://www.certik.com/）
- ChainSecurity Audit Reports（https://chainsecurity.com/smart-contract-audit-reports/）
- Cyfrin Audit Reports（https://github.com/Cyfrin/cyfrin-audit-reports）
- PeckShield Audit Reports（https://github.com/peckshield/publications）

- OpenZeppelin Audit Reports（https://blog.openzeppelin.com/tag/security-audits）
- **Complete List of Security Audit Reports ***

6. CTF 挑战

进行较有难度的 CTF 挑战以巩固知识。

- EtherHack（https://etherhack.positive.com/）
- SI Blockchain CTF（https://blockchain-ctf.securityinnovation.com/）
- QuillCTF（https://www.quillaudits.com/academy/ctf）
- Curta CTF（https://www.curta.wtf/）
- Paradigm CTF（https://ctf.paradigm.xyz/）
- Cipher Shastra CTF（https://ciphershastra.com/index.html）
- Damn Vulnerable DeFi（https://www.damnvulnerabledefi.xyz/）
- unhackedctf（https://github.com/unhackedctf）

融会贯通

在对头部 DeFi 的探索过程中将建立起对 DeFi 的深刻理解，接下来通过从底层 EVM 到 DeFi 上层经济模型的学习来继续加深对智能合约的理解。并且在此过程中，可以通过独立审计复杂智能合约以沉淀自己的审计方法论。

1. 深入了解 EVM

EVM 负责执行智能合约指令，深入了解 EVM 有助于我们对智能合约的部署、调用、执行、数据存储有更为深入的理解。同时可以为 Gas 优化、漏洞发现打好基础。

- 关于 EVM（https://www.evm.codes/about）
- The EVM From Scratch Book（https://evm-from-scratch.xyz/intro）
- **Noxx EVM 深入研究 ***

- Solidity 插槽数据解析（https://ethdebug.github.io/solidity-data-representation/）
- 以太坊黄皮书（https://ethereum.github.io/yellowpaper/paper.pdf）
  - 简单版（https://github.com/chronaeon/beigepaper）
- Quillhash EVM Mastery（https://github.com/Quillhash/EVM-Mastery）
- EVM 实现示例（https://github.com/noxx3xxon/evm-by-example）

2. Gas 优化设计

链上交易的执行都需要付出 Gas 成本。对于复杂合约来说，优化 Gas 可以降低用户交互成本，吸引用户使用。这就要求审计人员需要对 Gas 优化设计有一定的了解。

- Gas 优化参考 1（https://www.alchemy.com/overviews/solidity-gas-optimization）
- Gas 优化参考 2（https://www.rareskills.io/post/gas-optimization）
- Gas 优化参考 3

  （https://coinsbench.com/structs-in-solidity-best-practices-for-gas-efficiency-by-0xlazard-4e984a7485cf）

3. DeFi 经济模型

经济模型是 DeFi 产品的核心部分，所以了解经济模型的风险是很有必要的。在学习过程中应沉淀出自己的见解与方法论。

- 治理风险（https://arxiv.org/abs/2308.04267）
- DeFi 经济模型风险汇总（https://github.com/engn33r/DeFi-Risk-Modelling-Awesome）
  - Euler Oracle Manipulation Tool（https://oracle.euler.finance/）
  - **Chaos Lab Uniswap v3 Oracle Manipulation Risk ***
  - **Agent Buttercup simulation engine ***
  - Curve simulation tool（https://github.com/curveresearch/curvesim）
  - DELV agent-based simulation tool（https://github.com/delvtech/elf-simulations）
  - Uniswap v3 simulator **option 1 ***, **option 2 ***, **option 3 ***

4. 拆解分析复杂 DeFi 协议

在将这些技能融会贯通后，审计人员应具备拆解分析复杂的高原创性 DeFi 协议的能力。

- To be released...

5. 与同道者同行

学习其他优秀同道者所研究的内容可以给我们更多的启发，拓宽我们的视野。

- Bytes032（https://blog.bytes032.xyz/）
- Noxx（https://noxx.substack.com/）
- Mixbytes（https://mixbytes.io/blog/）
- Samczsun（https://samczsun.com/research/）
- Cmichel（https://cmichel.io/）
- Pessimistic（https://blog.pessimistic.io/）
- OfficerCia（https://officercia.mirror.xyz/）
- Smart Contract Research Forum（https://www.smartcontractresearch.org/）
- Zefram（https://zefram.xyz/posts/）
- Alin Tomescu（https://alinush.github.io/）
- Christoph Michel（https://cmichel.io/）
- Kyrian Alex（https://kyrianalex.substack.com/）
- ...

6. 快速应急分析

在独立审计过足够多的复杂项目，并经历过各种业务场景，沉淀了大量知识后，能够使我们快速应对突发安全事件并进行快速分析与输出。下面是一些常用的分析工具：

- 合约反编译工具
    - Dedaub（https://library.dedaub.com/decompile）

- Panoramix（https://github.com/palkeo/panoramix）
- abi-decompiler（https://github.com/Decurity/abi-decompiler）
- heimdall-rs（https://github.com/Jon-Becker/heimdall-rs）
- ethervm（https://ethervm.io/decompile）
- Pyevmasm（https://github.com/crytic/pyevmasm）

- 交易分析工具
  - Phalcon（https://explorer.phalcon.xyz/）
  - ethtx.info（https://ethtx.info/）
  - Tx eth samczsun（https://tx.eth.samczsun.com/）
  - Tenderly（https://tenderly.co/）
  - Sentio（https://app.sentio.xyz/explorer）
  - Eigenphi（https://eigenphi.io/）
  - SocketScan（https://socketscan.io/）
- Others

  Web3 Security Tools （https://github.com/Quillhash/Web3-Security-Tools）

  On Chain Investigations Tools List （https://github.com/OffcierCia/On-Chain-Investigations-Tools-List）

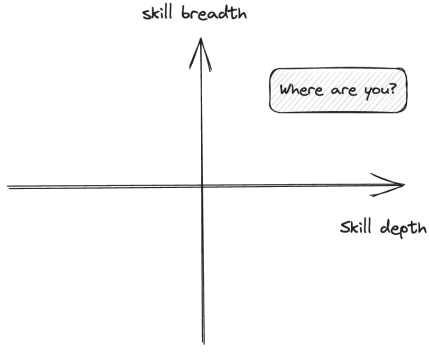
7. Bug Bounty 实战

进行实战，以最真实的场景检测能力。

- Immunefi （https://immunefi.com）
- BugRap （https://bugrap.io）
- Code4rena （https://code4rena.com）
- HackerOne （https://hackerone.com）
- HackenProof （https://hackenproof.com）
- HatsFinance （https://hats.finance）
- ...

破门而出

在由 Solidity 智能合约寻门而入后，不应再满足于在单一领域兜兜转转，而是应该沉淀出自己方法论，大胆地凿出一道新门，破门而出，在深耕当前领域的同时拓展其他领域。



1. 突破局限

在从 Solidity 智能合约入门后不应只局限于此，也应拓展其他类型、语言，并对其涉及的审计方法有所沉淀。

- 不只局限于 Solidity，Rust/Vyper/Cairo/Move 等智能合约语言也应了解
- 不只局限于智能合约，BTC/Cosmos/Solana/Starknet/EVM L2 等其他流行公链也应了解
- 不只局限于区块链，Web2.0/移动端等也应了解
- 深入了解密码学领域
- ...

2. 方法论

将智能合约安全审计技能融会贯通后，可以沉淀出属于自己的方法论，帮助我们快速的触及问题的核心并确定解决思路，好的方法论能让我们事半功倍。

- 对审计工作的方法论
- 对智能合约安全实践的方法论
- 做事的方法论
- 构建思维体系的方法论
- ...

3. 创造力

创造力是我们能够克敌制胜的法宝，是我们保持前进所需具备的东西。在按部就班地掌握技能后，再武装我们的思维，这能够使我们走得更远。

- 保持好奇心
  - 对新事物的敏感性
  - 不局限于自己的圈子/专业/职业
  - ...
- 追逐知识
  - 对知识保持敬畏
  - 探索新知识
- 黑客思维
  - 黑客也可以是一种精神也可以是一种思维，守正出奇
- 善于研究
  - 在进行研究时应有实际的结果/文档输出
- 工程化
  - 对于好点子，好的的研究成果应该善于工程化，并在实战中进行检验
    - SlowMist MistEye Monitoring System
    - SlowMist Contract Visibility Analysis Tool
    - SlowMist Static Vulnerability Scanner
    - ...

致谢

感谢朋友们对本文提出的宝贵建议。

- Cos（https://twitter.com/evilcos）
- 23pds （https://twitter.com/IM_23pds）
- T41nk （https://twitter.com/T41nk_）
- Doublenine
- Flush
- Blue
- Lizi

感谢 Jian 对英文版本的翻译工作，以及 Hik3 提供的封面图。

*

慢雾 (SlowMist) 区块链入门科普

https://mp.weixin.qq.com/mp/appmsgalbum?__biz=MzU4ODQ3NTM2OA==&action=getalbum&album_id=1378673890158936067&scene=126#wechat_redirect

OpenZeppelin Token

https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token

不同模式的代理合约介绍

https://ethereum-blockchain-developer.com/110-upgrade-smart-contracts/00-project/

OpenZeppelin Proxy

https://docs.openzeppelin.com/contracts/4.x/api/proxy

DevNet

https://docs.tenderly.co/devnets/intro-to-devnets

交易模拟

https://docs.tenderly.co/simulations-and-forks/intro-to-simulations

可视化的交易 Debug 工具

https://docs.tenderly.co/debugger/how-to-use-tenderly-debugger

用代码索引做在线 Debug

https://docs.sentio.xyz/sentio-debugger/code-insight

交易模拟中覆盖合约

https://docs.sentio.xyz/sentio-debugger/simulation#override-contract

Kaden: 智能合约攻击向量

https://github.com/kadenzipfel/smart-contract-vulnerabilities

RareSkills Smart Contract Security

https://www.rareskills.io/post/smart-contract-security

ConsenSys 智能合约最佳实践

https://github.com/ConsenSys/smart-contract-best-practices/blob/master/README-zh.md

Solidity 安全陷阱和最佳实践 101

https://secureum.substack.com/p/security-pitfalls-and-best-practices-101

Solidity 安全陷阱和最佳实践 201

https://secureum.substack.com/p/security-pitfalls-and-best-practices-201

DeFi 各类型介绍

https://teachyourselfcrypto.com/#ftoc-module-4-decentralized-finance-defi

基础的金融玩法

https://www.khanacademy.org/economics-finance-domain/core-finance/derivative-securities

Curve 牛顿迭代详解

https://0xreviews.xyz/posts/2022-02-28-curve-newton-method

SlowMist DeFi 被黑简析

https://docs.google.com/document/d/1b-uHJ7XDe1-xyaQQ9MYB3FGmYD7K_ULH8bUc20EZfu8/edit

DeFiHackLabs

https://web3sec.notion.site/web3sec/Web3-Security-ddaa8bf9a985494dbaf70d698345b899

Complete List of Security Audit Reports

https://github.com/0xNazgul/Blockchain-Security-Audit-List

Noxx EVM 深入研究

https://noxx.substack.com/p/evm-deep-dives-the-path-to-shadowy

Chaos Lab Uniswap v3 Oracle Manipulation Risk

https://community.chaoslabs.xyz/uniswap/twap

Agent Buttercup simulation engine

https://github.com/Cozy-Finance/agent-buttercup

Uniswap v3 simulator option 1

https://github.com/Bella-DeFinTech/uniswap-v3-simulator

Uniswap v3 simulator option 2

https://github.com/aloelabs/uniswap-simulator

Uniswap v3 simulator option 3

https://github.com/DefiLab-xyz/uniswap-v3-simulator

Learning Roadmap for Becoming a Smart Contract Auditor

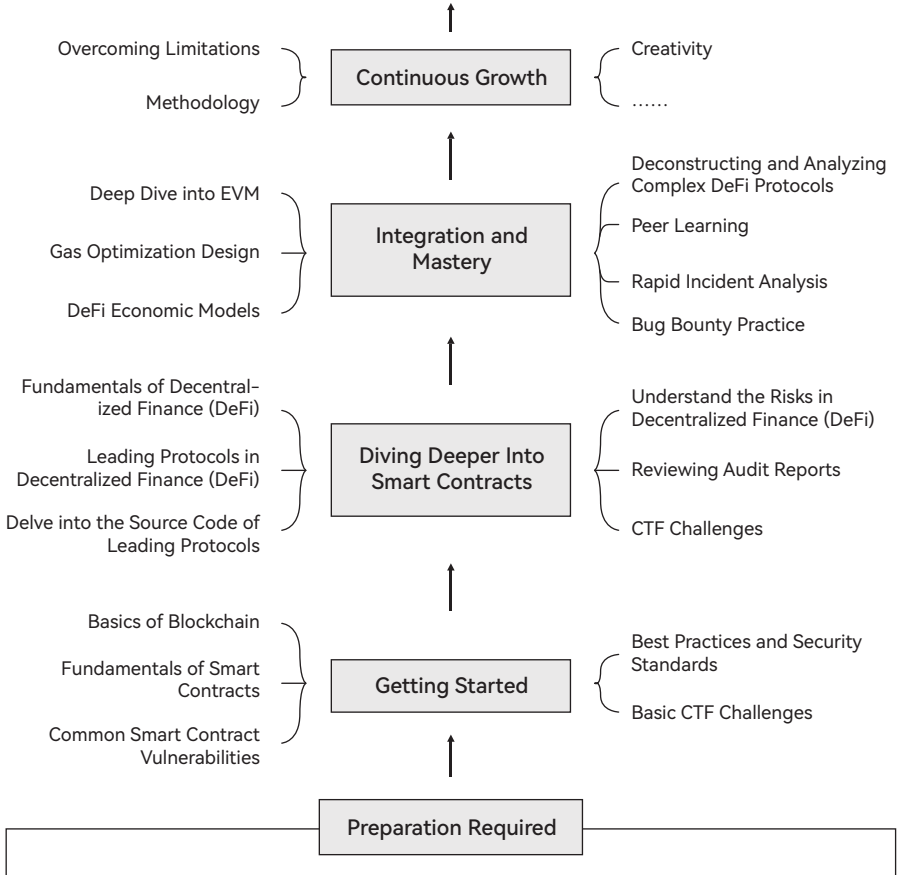
English Version

Introduction

This skill chart is a compilation of the abilities required for SlowMist's security team's smart contract security auditors. It aims to enumerate the essential skills needed for smart contract security audits and inspire team members to adopt a mindset of research, innovation, and engineering evolution.

Smart contract security auditing skills are primarily divided into four parts: Finding the Entrance, Singing by the Door, Integrative Mastery, and Breaking Through. These stages progressively outline the expertise needed at each level. However, before diving into these, it's crucial to equip our minds with some foundational skills, which will serve as our anchor point in the audit journey.

Roadmap



Unity of Knowledge and Action

- Learning should result in output, and output should be put into practice
- SlowMist Security Team's Knowledge Base
- Embracing the open-source culture of GitHub

Defend Fundamentals & Innovatively

- Sticking to the Basics
- Responsible Disclosure
- Hacker's Mentality

Team Consciousness

- Collaborative Audit Flow
- Audit Workflow
- Hacking Time Culture

Preparation Required

As Abraham Lincoln once said: "Give me six hours to chop down a tree and I will spend the first four sharpening the axe." This same approach can be applied to smart contract auditing. Strengthening our mindset before starting is essential, allowing us to move with conviction and travel further.

1. Unity of Knowledge and Action

Cognition and execution go hand in hand. Theoretical knowledge and its practical application should align seamlessly.

- Learning should result in output, and output should be put into practice.
- SlowMist Security Team's Knowledge Base (https://github.com/slowmist/Knowledge-Base)
- Embracing the open-source culture of GitHub

2. Defend Fundamentals & Innovatively

Ethics and the law are the foundational principles for security professionals. While staunchly adhering to these principles, security professionals must also forge strong technical skills, surprising adversaries when least expected.

- **Sticking to the Basics:**
  - Auditors should abide by the law and uphold ethical standards
- **Responsible Disclosure:**
  - SlowMist's alert procedure
  - FIRST's code of ethics (https://www.first.org/global/sigs/ethics/ethics-first)
- **Hacker's Mentality:**
  - While adhering to principles, prevail with unexpected strategies
  - Defend Fundamentals: Approach with reverence and always adhere to basic
  - Innovative: Think outside the box, be meticulous, employ reverse and open-minded thinking

3. Team Consciousness

The capabilities of an individual are always limited, but teamwork can effectively compensate for personal shortcomings.

- **Collaborative Audit Flow:**

  Collaborative auditing on the SlowMist MistPunk workbench ensures audit quality through technical means and accumulates auditing experience

- **Audit Workflow:**

  SlowMist's audit workflow ensures audit quality through managerial practices, serving to identify and fill in any gaps in the audit process

- **Hacking Time Culture:**

  Team members are encouraged to engage in spontaneous intellectual exchanges and sharing, aligning team capabilities through these collaborative interactions and thereby elevating the overall team proficiency

Getting Started

The development of the cryptographic world to date encompasses disciplines such as cryptography, economics, and data science. Faced with the immense volume of knowledge in the cryptographic world, determining a point of entry is crucial. In this phase, we will start by exploring Ethereum and its smart contract language, Solidity, as a gateway into the world of cryptocurrency.

1. Basics of Blockchain

Before understanding what smart contracts are, one should first comprehend the blockchain platform on which they operate.

- What is a blockchain? (https://www.investopedia.com/terms/b/blockchain.asp)
- Visual demonstration of blockchain (https://www.youtube.com/watch?v=_160oMzblY8)
- How cryptocurrencies work (https://www.youtube.com/watch?v=bBC-nXj3Ng4)
- Reading "Mastering Bitcoin" (https://github.com/bitcoinbook/bitcoinbook)
- Reading "Mastering Ethereum" (https://github.com/ethereumbook/ethereumbook)

  Emphasis should be on Chapters 1, 4, 5, 6, 7, and 13

2. Fundamentals of Smart Contracts

Different blockchains might employ various languages to develop smart contracts, such as Solidity, Move, Rust, Vyper, Cairo, C++, etc. Currently, Solidity remains the most popular and beginner-friendly smart contract language for EVM-compatible chains. It's essential to thoroughly read its language documentation. Moreover, one should understand the design standards of token contracts running on Ethereum and their specific contract implementations. Building on this foundation, it's crucial to understand how smart contracts can be made upgradable and to practically master the writing and testing of smart contracts.

Resources and Tools for Mastering Smart Contracts with Solidity.

- Solidity Official Documentation (https://docs.soliditylang.org/en/latest/)
- Essential Reading "Mastering Ethereum" (https://github.com/ethereumbook/ethereumbook)
  - Emphasis on reading the remaining chapters
- Understanding Ethereum's Standard Proposals (ERC) (https://eips.ethereum.org/erc)
  - ERC20 (https://eips.ethereum.org/EIPS/eip-20): Standard for fungible tokens
  - ERC165 (https://eips.ethereum.org/EIPS/eip-165): Interface standard
  - ERC173 (https://eips.ethereum.org/EIPS/eip-173): Contract ownership standard
  - ERC191 (https://eips.ethereum.org/EIPS/eip-191): Data signature standard
  - ERC601 (https://eips.ethereum.org/EIPS/eip-601): Deterministic wallet hierarchical structure standard
  - ERC721 (https://eips.ethereum.org/EIPS/eip-721): Non-fungible token standard
  - ERC777 (https://eips.ethereum.org/EIPS/eip-777): Interoperable token standard
  - ERC1155 (https://eips.ethereum.org/EIPS/eip-1155): Multi-token standard
  - ERC1167 (https://eips.ethereum.org/EIPS/eip-1167): Minimal proxy contract
  - ERC1967 (https://eips.ethereum.org/EIPS/eip-1967): Proxy data storage slots
  - ERC2612 (https://eips.ethereum.org/EIPS/eip-2612): Token permit signature
  - ERC4626 (https://eips.ethereum.org/EIPS/eip-4626): Token vault standard
- Studying OpenZeppelin's token implementations (https://github.com/OpenZeppelin/openzeppelin-contracts/tree/master/contracts/token)
- Understanding what upgradable contracts/proxy contracts are
  - Introduction to different proxy contract patterns (https://ethereum-blockchain-developer.com/110-upgrade-smart-contracts/00-project/)
  - Proxies Deep Dive (https://proxies.yacademy.dev/pages/proxies-list/)

    - OpenZeppelin Proxy implementation documentation (https://docs.openzeppelin.com/-contracts/4.x/api/proxy)
- Learning to Write Smart Contracts
    - WTF Solidity Smart Contract Tutorials (https://www.wtf.academy/en/)
    - Crypto Zombies (https://cryptozombies.io/en/course/)
    - Smart Contract Engineer (https://www.smartcontract.engineer/)
    - Solidity by Example (https://solidity-by-example.org/)
- Utilizing Smart Contract Build Tools
    - Popular Online IDEs
        - Remix (https://remix.ethereum.org/)
        - ChainIDE (https://chainide.com/)
        - Tenderly Sandbox (https://sandbox.tenderly.co/)
    - Familiarization with Package Managers
        - npm (https://www.npmjs.com/)
        - yarn (https://yarnpkg.com/)
        - pnpm (https://pnpm.io/)
    - Popular Smart Contract Testing and Debugging Frameworks
        - Foundry (https://book.getfoundry.sh/)
            - Convenient testing tools (https://book.getfoundry.sh/forge/tests)
            - Cheatcodes (https://book.getfoundry.sh/cheatcodes/)
            - Best practices (https://book.getfoundry.sh/tutorials/best-practices)
        - Hardhat (https://hardhat.org/hardhat-runner/docs/getting-started#overview)
            - Leveraging its potent plugins (https://hardhat.org/hardhat-runner/plugins)
        - Brownie (https://eth-brownie.readthedocs.io/en/stable/)
        - Tenderly (https://tenderly.co/)
            - Convenient DevNet development testing environment (https://docs.tenderly.co/devnets/intro-to-devnets)
            - Quick transaction simulations (https://docs.tenderly.co/simulations-and-forks/intro-to-simulations)
            - Visual transaction debugging tools (https://docs.tenderly.co/debugger/how-to-use-tenderly-debugger)
        - Sentio (https://app.sentio.xyz/explorer)
            - Online debugging with code insight (https://docs.sentio.xyz/sentio-debugger/code-insight)

        - Simulation with contract overrides (https://docs.sentio.xyz/sentio-debugger/-simulation#override-contract)
    - Interacting with Smart Contracts
        - Understanding JSON-RPC (https://ethereum.org/en/developers/docs/apis/json-rpc/)
        - ethers.js (https://docs.ethers.org/)
        - Web3.js (https://web3js.readthedocs.io/)
        - Web3.py (https://web3py.readthedocs.io/)
        - viem (https://viem.sh/)

3. Common Smart Contract Vulnerabilities

After mastering the fundamentals of smart contracts, it's essential to understand common vulnerabilities and the principles behind these vulnerabilities. Quillhash's vulnerability list, which aggregates multiple sources, offers a comprehensive view of the prevalent types of smart contract vulnerabilities. (For beginners, it's recommended to read through all the lists repeatedly to reinforce their understanding.)

- DASP Top 10 (https://www.dasp.co/)
- SWC Smart Contract Weakness Classification (https://swcregistry.io/)
- Smart Contract Security Guide (https://scsfg.io/hackers/)
- Kaden: Smart Contract Attack Vectors (https://github.com/kadenzipfel/smart-contract-vulnerabilities)
- Quillhash: Solidity Attack Vectors (https://github.com/Quillhash/Solidity-Attack-Vectors)
- RareSkills Smart Contract Security (https://www.rareskills.io/post/smart-contract-security)

4. Best Practices and Security Standards

As auditors, it's crucial to be familiar with the best practices and security standards of smart contracts. Best practices serve as a reference in identifying security issues during an audit, while security standards provide a basis for any security issues raised.

- Solidity Patterns (https://fravoll.github.io/solidity-patterns/)
- Solcurity (https://github.com/transmissions11/solcurity)
- ConsenSys Smart Contract Best Practices (https://github.com/Consensys/smart-contract-best-practices/blob/master/README.md)

- Solidity Security Pitfalls and Best Practices 101 (https://secureum.substack.com/p/security-pitfalls-and-best-practices-101)
- Solidity Security Pitfalls and Best Practices 201 (https://secureum.substack.com/p/security-pitfalls-and-best-practices-201)
- SCSVSv2 (https://github.com/securing/SCSVS/tree/prerelease/SCSVSv2)
- EEA EthTrust Certification (https://entethalliance.org/specs/ethtrust-sl/)
- Foundry Testing Best Practices (https://book.getfoundry.sh/tutorials/best-practices)

5. Basic CTF Challenges

After acquiring foundational knowledge about blockchain, smart contracts, and common vulnerabilities, it's beneficial to consolidate and apply this knowledge through basic Capture The Flag (CTF) challenges.

- OpenZeppelin Ethernaut (https://ethernaut.openzeppelin.com/)
- Capture the Ether (https://capturetheether.com/)

Diving Deeper Into Smart Contracts

Having grasped the basics of blockchain and smart contracts, we've essentially opened the door to Solidity smart contract security auditing. Beyond this door, the world of smart contracts is still vast. In this stage, we'll delve deeper into the expansive realm of smart contract applications, starting with decentralized finance (DeFi).

1. Fundamentals of Decentralized Finance (DeFi)

Blockchain and smart contracts have made the construction of DeFi possible, and the emergence of DeFi has spurred the rapid development of blockchains like Ethereum. Before delving further, one should understand: What is DeFi?

- Read "How To DeFi: Beginner" (https://landing.coingecko.com/how-to-defi/)
- Read "How To DeFi: Advanced" (https://landing.coingecko.com/how-to-defi/)
- Introduction to different types of DeFi (https://teachyourselfcrypto.com/#ftoc-module-4-decentralized-finance-defi)

    - Understand what each type of DeFi is
    - Grasp basic economic knowledge and common terminologies
- Introduction to fundamental financial strategies (https://www.khanacademy.org/economics-finance-domain/core-finance/derivative-securities)
- Economics Model 101 (https://tokenomicsdao.xyz/tokenomics101/)

2. Leading Protocols in Decentralized Finance (DeFi)

After an initial understanding of what DeFi is, it's important to delve deeper into what these platforms have achieved and how they achieved it. This can be accomplished by reviewing the technical documentation of the current leading DeFi protocols to gain preliminary insights into their implementations.

- MakerDAO (https://docs.makerdao.com/) (Collateralized Debt Position)
- AAVE (https://docs.aave.com/hub/) (Lending)
    - V2 (https://docs.aave.com/developers/v/2.0/)
    - V3 (https://docs.aave.com/developers/getting-started/readme)
- Compound (Lending)
    - V2 (https://docs.compound.finance/v2/)
    - V3 (https://docs.compound.finance/)
- Uniswap (Decentralized Exchange)
    - V2 (https://docs.uniswap.org/contracts/v2/overview)
    - V3 (https://docs.uniswap.org/contracts/v3/overview)
- Curve (Decentralized Exchange)
    - Technical Documentation (https://docs.curve.fi/)
    - Algorithm Overview (https://hackmd.io/@alltold/curve-magic)
    - Detailed Explanation of Curve's Newton Iteration (https://0xreviews.xyz/posts/2022-02-28-curve-newton-method)
- Chainlink (Oracle)
    - Price Oracles (https://docs.chain.link/data-feeds)
    - VRF (https://docs.chain.link/vrf/v2/introduction)(Verifiable Random Function)
- Convex Finance (Yield Optimization)
    - Protocol Overview (https://docs.convexfinance.com/convexfinance/)
    - Technical Documentation (https://docs.convexfinance.com/convexfinanceintegration/)

- Yearn Finance (Yield Aggregator)（https://docs.yearn.fi/getting-started/intro）
- GMX (Derivatives)（https://gmxio.gitbook.io/gmx/）
- Nexus Mutual (Insurance)（https://docs.nexusmutual.io/overview/）
- OpenSea (NFT Marketplace)（https://github.com/ProjectOpenSea/seaport#seaport）
- Set Protocol (Indexes)（https://docs.tokensets.com/）
- Lido (Liquid Staking)（https://docs.lido.fi/）
- ...and more

3. Delve into the Source Code of Leading Protocols

Many DeFi projects currently interrelate and combine, with some leading DeFi protocols forming the cornerstone of complex DeFi combinations. Hence, mastering the implementation of these DeFi initiatives is critical. After an initial understanding of DeFi's workings through protocol documentation, further comprehension can be achieved by studying the complete source code to grasp specific logic and economic models.

4. Understand the Risks in Decentralized Finance (DeFi)

DeFi isn't solely about smart contracts; the frontend and backend are vital components. Most users interact with DeFi through the frontend. Therefore, after understanding DeFi's operations and implementations, it's essential to learn and practice its risks through frontend security measures, backend security configuration requirements, and historical vulnerabilities in the DeFi sector.

- Understanding Web Frontend Security
  - Read the "SlowMist Web Frontend Security Best Practices Guide"
  - Dive deeper into "Web Frontend Hacker Techniques Revealed"
- Understanding DeFi Hacks
  - SlowMist DeFi Hack Analysis (https://docs.google.com/document/d/1b-uHJ7XDe1-xya-QQ9MYB3FGmYD7K_ULH8bUc20EZfu8/edit)
  - SlowMist Medium Articles (https://slowmist.medium.com/)
  - DeFiHackLabs (https://web3sec.notion.site/web3sec/Web3-Security-ddaa8bf9a985494dbaf70d698345b899)
  - Rekt (https://rekt.news/zh/)

- Immunefi（https://medium.com/@immunefi）
- QuillAudits（https://quillaudits.medium.com/）
- BlockSec（https://blocksecteam.medium.com/）
- Neptune Mutual（https://medium.com/@neptunemutual）
- PeckShield（https://twitter.com/peckshield）
- hacxyk（https://medium.com/@hacxyk）
- TrailOfBits（https://blog.trailofbits.com/）
- Secureum（https://secureum.substack.com/）
- Openzeppelin（https://blog.openzeppelin.com/security-audits/）
- OfferCIA（https://officercia.mirror.xyz/）

5. Reviewing Audit Reports

During an audit, an individual's perspective may miss certain aspects and cannot cover all scenarios. Therefore, reading other people's audit reports is crucial to learn different methods of vulnerability discovery and various auditing thought processes.

- SlowMist Audit Reports（https://github.com/slowmist/Knowledge-Base）
- Solodit Aggregation（https://solodit.xyz/）
- Code4rena Audit Reports（https://code4rena.com/reports）
- Consensys Audit Reports（https://consensys.net/diligence/audits/）
- QuillAudits Audit Reports（https://github.com/Quillhash/QuillAudit_Reports）
- Spearbit Audit Reports（https://github.com/spearbit/portfolio/tree/master/pdfs）
- Sherlock Audit Reports（https://github.com/sherlock-protocol/sherlock-reports）
- ADBK Audit Reports（https://github.com/abdk-consulting/audits）
- BlockSec Audit Reports（https://github.com/blocksecteam/audit-reports）
- Certik Audit Reports（https://www.certik.com/）
- ChainSecurity Audit Reports（https://chainsecurity.com/smart-contract-audit-reports/）
- Cyfrin Audit Reports（https://github.com/Cyfrin/cyfrin-audit-reports）
- PeckShield Audit Reports（https://github.com/peckshield/publications）
- OpenZeppelin Audit Reports（https://blog.openzeppelin.com/tag/security-audits）
- Complete List of Security Audit Reports (https://github.com/0xNazgul/Blockchain-Security-Audit-List)

6. CTF Challenges

Engage in more advanced CTF challenges to test new skills and grow.

- EtherHack（https://etherhack.positive.com/）
- SI Blockchain CTF（https://blockchain-ctf.securityinnovation.com/）
- QuillCTF（https://www.quillaudits.com/academy/ctf）
- Curta CTF（https://www.curta.wtf/）
- Paradigm CTF（https://ctf.paradigm.xyz/）
- Cipher Shastra CTF（https://ciphershastra.com/index.html）
- Damn Vulnerable DeFi（https://www.damnvulnerabledefi.xyz/）
- unhackedctf（https://github.com/unhackedctf）

Integration and Mastery

Through the exploration of leading DeFi platforms, a profound understanding of DeFi will be established. Moving forward, by learning from the foundational layer of EVM to the upper economic models of DeFi, we can continue to deepen our grasp on smart contracts. During this process, independently auditing complex smart contracts can help solidify one's own audit methodology.

1. Deep Dive into EVM

The EVM (Ethereum Virtual Machine) is responsible for executing smart contract instructions. A comprehensive understanding of the EVM aids in a more in-depth grasp of the deployment, invocation, execution, and data storage of smart contracts. This foundational knowledge is also pivotal for Gas optimization and discovering vulnerabilities.

- About EVM (https://www.evm.codes/about)
- The EVM From Scratch Book (https://evm-from-scratch.xyz/intro)
- Noxx's In-depth Research on EVM (https://noxx.sub-stack.com/p/evm-deep-dives-the-path-to-shadowy)

- Parsing Solidity Slot Data (https://ethdebug.github.io/solidity-data-representation/)
- Ethereum Yellow Paper (https://ethereum.github.io/yellowpaper/paper.pdf)
  - Simplified Version (https://github.com/chronaeon/beigepaper)
- Quillhash EVM Mastery (https://github.com/Quillhash/EVM-Mastery)
- EVM Implementation Examples (https://github.com/noxx3xxon/evm-by-example)

2. Gas Optimization Design

All on-chain transactions incur Gas costs. For complex contracts, optimizing Gas can reduce user interaction costs, thereby attracting more users. This demands that auditors have a certain understanding of Gas optimization design.

- Gas Optimization References 1 (https://www.alchemy.com/overviews/solidity-gas-optimization)
- Gas Optimization References 2 (https://www.rareskills.io/post/gas-optimization)
- Gas Optimization References 3 (https://coinsbench.com/structs-in-solidity-best-practices-for-gas-efficiency-by-0xlazard-4e984a7485cf)

3. DeFi Economic Models

The economic model is a core component of DeFi products, so it's essential to understand the risks associated with these models. Throughout the learning process, one should develop and consolidate their perspectives and methodologies.

- Governance Risks (https://arxiv.org/abs/2308.04267)
- Summary of Risks in DeFi Economic Models (https://github.com/engn33r/DeFi-Risk-Modelling-Awesome)
  - Euler Oracle Manipulation Tool (https://oracle.euler.finance/)
  - Chaos Lab Uniswap v3 Oracle Manipulation Risk (https://community.chaoslabs.xyz/uniswap/twap)
  - Agent Buttercup simulation engine (https://github.com/Cozy-Finance/agent-buttercup)
  - Curve simulation tool (https://github.com/curveresearch/curvesim)
  - DELV agent-based simulation tool (https://github.com/delvtech/elf-simulations)
  - Uniswap v3 simulator option 1, option 2, option 3

(https://github.com/Bella-DeFinTech/uniswap-v3-simulator)
(https://github.com/aloelabs/uniswap-simulator)
(https://github.com/DefiLab-xyz/uniswap-v3-simulator)

4. Deconstructing and Analyzing Complex DeFi Protocols

After mastering these skills, auditors should be capable of dissecting and analyzing complex, highly innovative DeFi protocols.

- To be released...

5. Peer Learning

Learning from other outstanding peers who are researching various topics can provide us with more insights and broaden our horizons.

- Bytes032（https://blog.bytes032.xyz/）
- Noxx（https://noxx.substack.com/）
- Mixbytes（https://mixbytes.io/blog/）
- Samczsun（https://samczsun.com/research/）
- Cmichel（https://cmichel.io/）
- Pessimistic（https://blog.pessimistic.io/）
- OfficerCia（https://officercia.mirror.xyz/）
- Smart Contract Research Forum（https://www.smartcontractresearch.org/）
- Zefram（https://zefram.xyz/posts/）
- Alin Tomescu（https://alinush.github.io/）
- Christoph Michel（https://cmichel.io/）
- Kyrian Alex（https://kyrianalex.substack.com/）
- And more

6. Rapid Incident Analysis

After independently auditing numerous complex projects, accumulating extensive knowledge, and experiencing various business scenarios, auditors should be able to respond quickly to unforeseen security incidents and conduct rapid analysis and reporting. Here are some commonly used analysis tools:

- Contract Decompilation Tools
  - Dedaub (https://library.dedaub.com/decompile)
  - Panoramix (https://github.com/palkeo/panoramix)
  - abi-decompiler (https://github.com/Decurity/abi-decompiler)
  - heimdall-rs (https://github.com/Jon-Becker/heimdall-rs)
  - ethervm (https://ethervm.io/decompile)
  - Pyevmasm (https://github.com/crytic/pyevmasm)
- Transaction Analysis Tools
  - Phalcon (https://explorer.phalcon.xyz/)
  - ethtx.info (https://ethtx.info/)
  - Tx eth samczsun (https://tx.eth.samczsun.com/)
  - Tenderly (https://tenderly.co/)
  - Sentio (https://app.sentio.xyz/explorer)
  - Eigenphi (https://eigenphi.io/)
  - SocketScan (https://socketscan.io/)
- Others

  Web3 Security Tools (https://github.com/Quillhash/Web3-Security-Tools)

  On Chain Investigations Tools List (https://github.com/OffcierCia/On-Chain-Investigations-Tools-List)

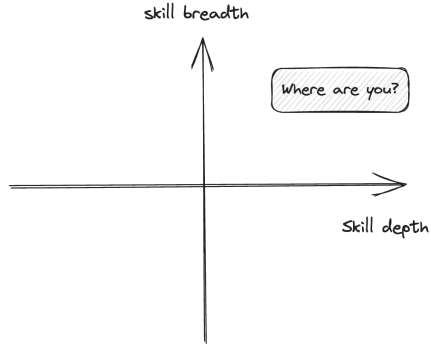
7. Bug Bounty Practice

Engaging in real-world scenarios is the best way to test your skills.

- Immunefi （https://immunefi.com）
- BugRap （https://bugrap.io）
- Code4rena （https://code4rena.com）
- HackerOne （https://hackerone.com）
- HackenProof （https://hackenproof.com）
- HatsFinance （https://hats.finance）
- And more

Continuous Growth

After entering the world of Solidity smart contracts, one should not be content with circling within a single domain. Instead, you should crystallize your own methodology, boldly carve out a new path, break through, and expand into other areas while deepening your expertise in the current field.



1. Overcoming Limitations

- Don't confine yourself to Solidity; explore other smart contract languages like Rust, Vyper, Cairo, and Move
- Don't limit yourself to smart contracts; understand popular public blockchains like BTC, Cosmos, Solana, Starknet, EVM L2, and more
- Beyond blockchain, gain insights into areas like Web2.0 and mobile development
- Dive deeper into the field of cryptography
- And more

2. Methodology

After mastering the skills of smart contract security auditing, you can develop your own methodology. This helps you quickly identify the core issues and determine problem-solving approaches. A good methodology can greatly increase your efficiency.

- Methodology for auditing work
- Methodology for smart contract security practices
- Approaches to problem-solving
- Building a thinking framework

3. Creativity

Creativity is the weapon that allows us to conquer challenges and the essential attribute for progress. After acquiring skills systematically, nurture your creativity to advance further.

- Cultivate curiosity
  - Learn new things
  - Don't limit yourself to your circle, profession, or field
- Pursue knowledge relentlessly
  - Approach knowledge with reverence
  - Explore new realms of knowledge
- Embrace the hacker mindset
  - Staying true to principles while thinking outside the box
- Be adept at research
  - Aim for practical results and publish research
- Implement engineering principles
  - Turn ideas and research into practical solutions and test them in real-world
    - SlowMist MistEye Monitoring System
    - SlowMist Contract Visibility Analysis Tool
    - SlowMist Static Vulnerability Scanner
    - And more

Acknowledgments

Special thanks to friends who provided valuable feedback:

- Cos（https://twitter.com/evilcos）
- 23pds （https://twitter.com/IM_23pds）
- T41nk （https://twitter.com/T41nk_）
- Doublenine
- Flush
- Blue
- Lizi

Thanks to Jian for the English translation and to Hik3 for designing the cover image.

3

基于区块链的加密货币安全审计指南

Blockchain-Based Cryptocurrency Security Audit Guide

Johan@SlowMist Team

SLOWMIST

https://github.com/slowmist/Cryptocurrency-Security-Audit-Guide

基于区块链的加密货币安全审计指南

中文版

1. 加密货币威胁建模

SlowMist 使用多种模型来识别加密货币系统存在的威胁。

CIA 三元组：保密性、完整性和可用性性 (Confidentiality, Integrity and Availability)，这是安全基础架构中主要的安全目标和宗旨。

STRIDE 模型：Spoofing（欺骗）、Tampering（篡改）、Repudiation（否认）、Information Disclosure（信息泄露）、Denial of Service（拒绝服务）、Elevation of Privilege（提权）。

DREAD 模型：Damage potential（潜在损害）、Reproducibility（可重现性）、Exploitability（可利用性）、Affected users（受影响用户）、Discoverability（可发现性）。

PASTA：定义目标、定义技术范围、分解应用程序、分析威胁、识别漏洞、列举攻击、分析风险和影响。

2. 测试方法

测试方法如下:

测试方法	描述
黑盒测试	黑盒测试从用户角度检查程序，通过提供多种输入场景并检查输出。黑盒测试人员没有内部代码的访问权限。系统交付前进行的最终验收测试是黑盒测试的一个常见例子。
灰盒测试	灰盒测试结合了两种方法，在软件验证中很受欢迎。在这种方法中，测试人员从用户角度检查软件，分析输入和输出。他们还可以访问源代码，并利用它来帮助设计测试。然而，他们在测试期间不会分析程序的内部工作。
白盒测试	白盒测试检查程序的内部逻辑结构，逐行分析代码，查找潜在错误。

在黑盒测试和灰盒测试中，我们使用模糊测试、脚本测试等方法，通过提供随机数据或构建特定结构的数据来测试接口或组件的鲁棒性，挖掘一些边界条件下的系统异常行为，如 bug 或性能异常。在白盒测试中，我们通过代码审查等方法分析代码的对象定义和逻辑实现，结合安全团队在已知区块链安全漏洞上的相关经验，确保代码中的关键逻辑和关键组件没有已知漏洞；同时，进入新场景和新技术的漏洞挖掘模式，发现可能的 0day 错误。

3. 漏洞严重性

通用漏洞评分系统（CVSS）是一个用于传达软件漏洞特征和严重性的开放框架。CVSS 由三个指标组组成：基础、时间和环境。基础组代表漏洞的内在质量，这些质量在时间和用户环境中是恒定的；时间组反映漏洞随时间变化的特征；环境组代表用户环境中独特的漏洞特征。基础

指标生成的分数范围为 0 到 10，然后可以通过对时间和环境指标评分进行修改。CVSS 分数还表示为矢量字符串，矢量字符串是用于导出分数的值的压缩文本表示。

根据 CVSS 方法，SlowMist 团队开发了区块链漏洞严重性级别，如下所示：

级别	描述
严重	严重级别漏洞将对区块链项目的安全产生重大影响，强烈建议修复严重漏洞。
高	高级别漏洞将影响区块链项目的正常运行，强烈建议修复高风险漏洞。
中	中级别漏洞将影响区块链项目的运行，建议修复中风险漏洞。
低	低级别漏洞可能在某些场景下影响区块链项目的运行，建议项目方评估并考虑是否需要修复这些漏洞。
弱点	理论上存在安全风险，但在工程上极难重现。
建议	有更好的编码或架构实践。
信息	一些符合项目设计意图，但可能会导致用户资产损失的特性。

4. 公链安全研究

SlowMist 团队的**区块链威胁情报***系统持续追踪正在发生的安全事件，并将威胁情报应用于安全咨询与审计服务中。

SlowMist 团队对公开已知的区块链安全漏洞进行分析研究，汇编整理出了**区块链常见漏洞列表***。

5. 公链安全审计

SlowMist 团队的公链安全审计综合使用了黑盒、灰盒、白盒三种测试方法，同时根据审计需求不同，推出了以黑灰盒审计为主的主网安全审计、layer2 安全审计，和以白盒审计为主的源代码安全审计，同时为一些开发框架定制应用链安全审计方案。

5.1 主网 & layer2 项目安全审计

在主网 & layer2 项目安全审计中，SlowMist 团队采用“黑盒 + 灰盒”策略，以最接近真实攻击的方式对项目进行快速的安全测试。

SlowMist 团队检查的漏洞包括：

- 私钥随机数熵不足
- 私钥种子转换精度损失
- 对称加密算法的理论可靠性评估
- 对称加密算法依赖库的供应链安全
- 密钥库加密强度检测
- 哈希算法长度扩展攻击
- 哈希算法的理论可靠性评估
- 签名算法的理论可靠性评估
- secp256k1 k 值随机性安全
- secp256k1 r 值重用私钥提取攻击
- ECC 签名的可塑性攻击
- ed25519 私钥提取攻击
- Schnorr 私钥提取攻击
- ECC 曲线攻击
- Merkle 树可塑性攻击（CVE-2012-2459）
- 原生特性虚假充值
- 基于合约调用的虚假充值
- 原生链交易重放攻击
- 跨链交易重放攻击
- 交易锁定攻击
- 交易费用未动态调整
- RPC 远程密钥盗窃攻击
- RPC 端口可识别性
- RPC 开放跨域漏洞导致本地钓鱼攻击
- JsonRPC 畸形包拒绝服务攻击
- RPC 数据库注入
- RPC 通信加密
- 过度的管理员权限
- 非隐私/非暗币审计
- 核心节点数量不足

- 核心节点物理位置过度集中
- P2P 节点最大连接数限制
- P2P 节点独立IP连接限制
- P2P 入站/出站连接限制
- P2P 变形攻击
- P2P 通信加密
- P2P 端口可识别性
- 共识算法潜在风险评估
- 区块时间偏移攻击
- 矿工磨矿攻击
- PoS/BFT 双重签名惩罚

5.2 源代码安全审计

源代码安全审计是指 SlowMist 团队采用“白盒”策略，对项目的相关源代码进行最全面的安全测试。白盒审计通常需要结合自动化静态代码分析和人工手动分析两种形式。

5.2.1 静态源代码分析

SlowMist 团队使用开源或商业代码扫描工具对代码进行静态扫描，并人工解析发现的问题，我们支持所有流行语言，如 C/C++/Golang/Rust/Java/Nodejs/C#

SlowMist 团队检查的静态编码问题有：

- 未使用的变量或导入 - 声明但未使用的变量或导入模块。
- 代码格式问题 - 缩进不一致、行长度过长等。
- 资源未正确关闭 - 如文件、数据库连接等未关闭。
- 魔法数字 - 直接使用数字常量而非命名常量。
- 潜在的安全漏洞 - 如SQL注入、XSS等安全隐患。
- 整数溢出 - 当计算结果超出整数类型的范围时可能导致意外行为。
- 浮点数精度问题 - 由于浮点数表示的限制，可能导致计算误差。
- 死锁 - 多线程编程中，线程互相等待对方释放资源而陷入僵局。
- 竞态条件 - 多线程或并发环境下，程序的行为依赖于不可控的执行顺序。

- 内存泄漏 - 动态分配的内存未被正确释放，导致程序占用的内存持续增加。
- 无限递归 - 递归函数没有正确的终止条件，导致栈溢出。
- 字符串格式化漏洞 - 不安全的字符串格式化可能导致安全问题。
- 除零错误 - 在除法运算中未检查除数是否为零。
- 空指针解引用 - 试图访问空指针指向的内存位置。
- 缓冲区溢出 - 向缓冲区写入超出其容量的数据，可能导致安全漏洞。
- 类型转换错误 - 不当的类型转换可能导致数据丢失或不正确的结果。
- 硬编码密钥或敏感信息 - 将密钥或敏感信息直接写入代码中，可能导致安全风险。
- 代码复杂度过高 - 函数或方法过长，逻辑分支过多。
- 代码重复 - 相同或相似的代码段在多处出现。
- 命名不规范 - 变量、函数、类等命名不清晰或不一致。
- 注释不足或过时 - 缺乏必要的注释，或注释与代码不符。
- 耦合度高 - 模块间依赖关系复杂，难以维护和扩展。
- 低内聚 - 模块或类的功能不够集中，职责不明确。
- 异常处理不当 - 捕获过于宽泛的异常，或忽略异常。
- 硬编码 - 直接在代码中使用常量值，而非配置参数。
- 代码格式不一致 - 缩进、空格使用等不统一。
- 性能问题 - 如不必要的循环、频繁的对象创建等。
- 可测试性差 - 代码难以进行单元测试或集成测试。
- 违反设计原则 - 如单一职责原则、开闭原则等。
- 可读性差 - 代码结构混乱，难以理解。
- 不安全的随机数生成 - 使用不适合安全用途的随机数生成方法。
- 时间和状态问题 - 如TOCTOU（Time-of-check to time-of-use）漏洞。
- 路径遍历 - 未正确验证文件路径，可能导致访问未授权的文件。
- 依赖库过时 - 引入的库已失去维护或存在安全漏洞。

5.2.2 手动代码审查

SlowMist 团队逐行检查代码，查找编码缺陷和逻辑错误，我们关注的漏洞范围主要包括：

- 加密签名安全
- 账号与交易安全
- RPC 安全
- P2P 安全
- 共识安全
- 业务逻辑安全

5.3 应用链安全审计

SlowMist 团队采用“白盒”策略，对项目进行全面的安全测试，寻找常见的编码陷阱，如：

- 重放漏洞
- 重新排序漏洞
- 竞态条件漏洞
- 权限控制漏洞
- 块数据依赖漏洞
- 函数显式可见性
- 算术精度偏差漏洞
- 恶意事件日志
- 异步调用安全

目前我们支持：

1. 基于 Cosmos-SDK 框架的区块链审计
2. 基于 Substrate 框架的区块链审计

6. 区块链应用审计

6.1 智能合约安全审计

1. **Ethereum(Solidity) 智能合约安全审计 ***
2. **EOS(C++) 智能合约安全最佳实践 ***
3. **Solana(Rust) 智能合约安全最佳实践 ***
4. Near 智能合约安全审计
5. Move 智能合约安全审计
6. Ton 智能合约安全审计

6.2 其他应用

1. 零知识电路安全审计
2. 跨链桥应用安全审计
3. 浏览器插件钱包安全审计
4. 交易所安全审计

*

区块链威胁情报

https://bti.slowmist.com/

区块链常见漏洞列表

https://github.com/slowmist/Cryptocurrency-Security-Audit-Guide/blob/main/Blockchain-Common-Vulnerability-List.md

Ethereum(Solidity) 智能合约安全审计

https://www.slowmist.com/en/service-smart-contract-security-audit.html

EOS(C++) 智能合约安全最佳实践

https://github.com/slowmist/eos-smart-contract-security-best-practices

Solana(Rust) 智能合约安全最佳实践

https://github.com/slowmist/solana-smart-contract-security-best-practices

Blockchain-Based Cryptocurrency Security Audit Guide

English Version

1. Cryptocurrency Threat Modeling

SlowMist uses multiple models to identify cryptocurrency threats.

CIA Triad:

Confidentiality, Integrity, and Availability. These are the primary security goals and principles in a security infrastructure.

STRIDE Model:

Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service (DoS), and Elevation of Privilege.

DREAD Model:

Damage Potential, Reproducibility, Exploitability, Affected Users, and Discoverability.

PASTA:

Define objectives, define the technical scope, decompose the application, analyze threats, identify vulnerabilities, enumerate attacks, and analyze risks and impacts.

2. Testing Method

The testing methods are as follows:

Test Method	Description
Black-box testing	Black-box testing examines the program from a user perspective by providing a wide variety of input scenarios and inspecting the output. Black-box testers do not have access to the internal code. Final acceptance testing that occurs prior to system delivery is a common example of black-box testing.
Gray-box testing	Gray-box testing combines the two approaches and is popular for software validation. In this approach, testers examine the software from a user perspective, analyzing inputs and outputs. They also have access to the source code and use it to help design their tests. They do not, however, analyze the inner workings of the program during their testing.
White-box testing	White-box testing examines the internal logical structures of a program and steps through the code line by line, analyzing the program for potential errors.

In black-box testing and gray-box testing, we use fuzz testing, script testing and other methods to test the robustness of interfaces or components by feeding random data or constructing data with a specific structure, and to mine some boundaries. Abnormal behavior of the system under conditions such as bugs or abnormal performance. In the white-box test, we analyze the object definition and logic implementation of the code through methods such as code review, combined with the relevant experience accumulated by the security team on known blockchain security vulnerabilities, to ensure that the key logic and key components in the code are correct. Achieve no known vulnerabilities; at the same time, enter the vulnerability mining mode for new scenarios and new technologies, and find possible 0day errors.

3. Vulnerability Severity

The Common Vulnerability Scoring System (CVSS) is an open framework for communicating the characteristics and severity of software vulnerabilities. CVSS consists of three metric groups: Base, Temporal, and Environmental. The Base group represents the intrinsic qualities of a vulnerability that are constant over time and across user environments, the Temporal group reflects the characteristics of a vulnerability that change over time, and the Environmental group represents the characteristics of a vulnerability that are unique to a user's environment. The Base metrics produce a score ranging from 0 to 10, which can then be modified by scoring the Temporal and Environmental metrics. A CVSS score is also represented as a vector string, a compressed textual representation of the values used to derive the score.

According to the CVSS method, SlowMist team develops the blockchain vulnerability severity level, they are as follows:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the blockchain project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the blockchain project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the blockchain project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the blockchain project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.

Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.
Infomation	Some features that align with the project design intentions but may lead to user asset loss.

4. Public Blockchain Security Research

The SlowMist team's Blockchain Threat Intelligence (https://bti.slowmist.com/) system continuously tracks ongoing security incidents and applies threat intelligence to its security audit services.

The SlowMist team analyzes and studies publicly known blockchain security vulnerabilities and has compiled a Blockchain Common Vulnerability List (https://github.com/slowmist/Cryptocurrency-Security-Audit-Guide/blob/main/Blockchain-Common-Vulnerability-List.md).

5. Public Blockchain Security Audit

The SlowMist team employs a comprehensive approach utilizing black-box, gray-box, and white-box testing methods. Depending on the specific audit requirements, they offer various services including a mainnet security audit and layer 2 security audit that primarily use black-box and gray-box methods, and a source code security audit that focuses on white-box testing. Additionally, they provide customized application chain security audit solutions tailored for specific development frameworks.

5.1 Mainnet & layer2 project security audits

In mainnet & layer2 project security audits, the SlowMist team employs a "black-box + gray-box" strategy to conduct rapid security testing in a manner that closely simulates real-world attacks.

The vulnerabilities checked by the SlowMist team include:

- Insufficient entropy of private key random numbers
- Precision loss in private key seed conversion
- Theoretical reliability assessment of symmetric encryption algorithms
- Supply chain security of symmetric crypto algorithm reference libraries
- Keystore encryption strength detection
- Hash algorithm length extension attack
- Theoretical reliability assessment of hash algorithms
- Theoretical reliability assessment of signature algorithms
- secp256k1 k-value randomness security
- secp256k1 r-value reuse private key extraction attack
- ECC signature malleability attack
- ed25519 private key extraction attack
- Schnorr private key extraction attack
- ECC twist attack
- Merkle-tree Malleability attack (CVE-2012-2459)
- Native characteristic false recharge
- Contract call-based false recharge
- Native chain transaction replay attack
- Cross-chain transaction replay attack
- Transaction lock attack
- Transaction fees not dynamically adjusted
- RPC remote key theft attack
- RPC port identifiability
- RPC open cross-domain vulnerability to local phishing attacks
- JsonRPC malformed packet denial-of-service attack
- RPC database injection
- RPC communication encryption
- Excessive administrator privileges
- Non-privacy/Non-dark Coin Audit
- Insufficient number of core nodes
- Excessive concentration of core node physical locations
- P2P node maximum connection limit

- P2P node independent IP connection limit
- P2P inbound/outbound connection limit
- P2P shapeshift attack
- P2P communication encryption
- P2P port identifiability
- Consensus algorithm potential risk assessment
- Block time offset attack
- Miner grinding attack
- PoS/BFT double-signing penalty

5.2 Code-based Testing Audit

Source code security auditing refers to the comprehensive security testing of a project's source code by the SlowMist team using a "**white-box**" strategy. White-box auditing typically involves a combination of automated static code analysis and manual review.

5.2.1 Static Source Code Analysis

The SlowMist team utilizes open-source or commercial code scanning tools for static code analysis and manually examines the identified issues. We support all popular languages, including **C/C++/Golang/Rust/Java/Nodejs/C#**.

The static coding issues checked by the SlowMist team include:

- Unused Variables or Imports: Declared but unused variables or imported modules.
- Code Formatting Issues: Inconsistent indentation, excessive line length, etc.
- Improper Resource Closure: Unclosed resources such as files or database connections.
- Magic Numbers: Direct use of numeric constants instead of named constants.
- Potential Security Vulnerabilities: Issues like SQL injection, XSS, etc.
- Integer Overflow: Unexpected behavior due to calculations exceeding the range of integer types.
- Floating-Point Precision Issues: Calculation errors due to the limitations of floating-point representation.
- Deadlocks: Threads waiting on each other to release resources, causing a stalemate.

- Race Conditions: Program behavior depending on the uncontrolled order of execution in multi-threaded or concurrent environments.
- Memory Leaks: Dynamically allocated memory not properly released, leading to increasing memory usage.
- Infinite Recursion: Recursive functions lacking proper termination conditions, causing stack overflow.
- String Formatting Vulnerabilities: Insecure string formatting leading to potential security issues.
- Divide-by-Zero Errors: Failure to check for zero in divisor during division operations.
- Null Pointer Dereferencing: Attempting to access memory at the location pointed to by a null pointer.
- Buffer Overflow: Writing data exceeding the buffer's capacity, potentially leading to security vulnerabilities.
- Type Conversion Errors: Improper type conversions causing data loss or incorrect results.
- Hard-Coded Keys or Sensitive Information: Directly embedding keys or sensitive information in code, posing security risks.
- High Code Complexity: Long functions or methods, excessive logical branches.
- Code Duplication: Identical or similar code blocks appearing in multiple locations.
- Inconsistent Naming: Unclear or inconsistent naming of variables, functions, classes, etc.
- Insufficient or Outdated Comments: Lack of necessary comments or comments that do not match the code.
- High Coupling: Complex interdependencies between modules, making maintenance and expansion difficult.
- Low Cohesion: Modules or classes with functions that are not closely related, with unclear responsibilities.
- Improper Exception Handling: Catching overly broad exceptions or ignoring exceptions.
- Hard-Coding: Using constant values directly in the code instead of configuration parameters.
- Inconsistent Code Formatting: Non-uniform use of indentation, spaces, etc.
- Performance Issues: Unnecessary loops, frequent object creation, etc.
- Poor Testability: Code that is difficult to unit test or integrate test.
- Violation of Design Principles: Violations of principles like the Single Responsibility Principle, Open/Closed Principle, etc.
- Poor Readability: Confusing code structure, making it difficult to understand.

- Insecure Random Number Generation: Using random number generation methods unsuitable for secure purposes.
- Time and State Issues: Vulnerabilities like TOCTOU (Time-of-Check to Time-of-Use).
- Path Traversal: Failing to properly validate file paths, potentially accessing unauthorized files.
- Outdated Dependencies: Use of libraries that are no longer maintained or have known security vulnerabilities.

5.2.2 Manual Code Review

The SlowMist team performs a line-by-line code review to identify coding flaws and logical errors. The vulnerabilities we focus on mainly include:

- Cryptographic signature security
- Account and transaction security
- RPC security
- P2P security
- Consensus security
- Business logic security

5.3. Application Chain Security Audit

The SlowMist team adopts the strategy of "**White-box**" to conduct a complete security test on the project, looking for common coding pitfalls as follows:

- Replay Vulnerability
- Reordering Vulnerability
- Race Conditions Vulnerability
- Authority Control Vulnerability
- Block data Dependence Vulnerability
- Explicit Visibility of Functions
- Arithmetic Accuracy Deviation Vulnerability
- Malicious Event Log
- Asynchronous Call Security

Currently we support:

1. Cosmos-SDK Framework Based Blockchain Audit
2. Substrate Framework Based Blockchain Audit

6. Blockchain Application Audit

6.1 Smart Contract Security Audit

1. Ethereum(Solidity) Smart Contract Security Audit (https://www.slowmist.com/en/service-smart-contract-security-audit.html)
2. EOS(C++) Smart Contract Security Best Practices (https://github.com/slowmist/eos-smart-contract-security-best-practices)
3. Solana(Rust) Smart Contract Security Best Practices (https://github.com/slowmist/solana-smart-contract-security-best-practices)
4. NEAR smart contract security audit
5. Move smart contract security audit
6. TON smart contract security audit

6.2 Other Application

1. Zero-Knowledge Circuit Security Audit
2. Interchain Bridge Application Security Audit
3. Browser Plugin Wallet Security Audit
4. Exchange Security Audit

4

加密资产安全解决方案

Cryptocurrency Security Solution by SlowMist

Blue@SlowMist Team

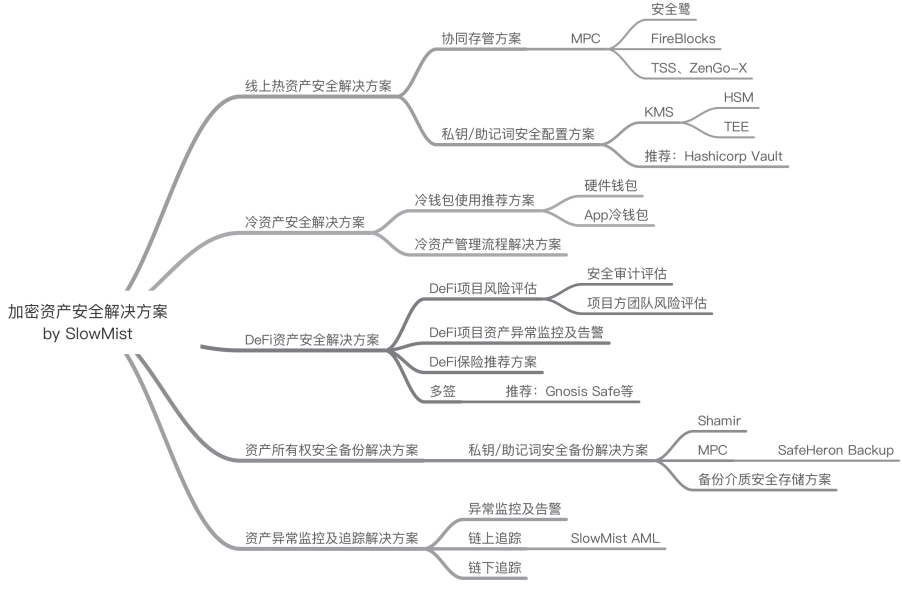
SLOWMIST

https://github.com/slowmist/cryptocurrency-security

加密资产安全解决方案

中文版

本方案由慢雾安全团队多年第一线服务甲方的实践经验沉淀，旨在为加密世界的参与者提供全方位的资产安全解决方案。我们将加密资产安全整理划分为了以下五大部分，并针对每一部分做了详细的解读，包括各类风险及相关的解决方案。



- **线上热资产安全解决方案**
- **冷资产安全解决方案**
- **DeFi 资产安全解决方案**
- **资产所有权安全备份解决方案**
- **资产异常监控及追踪解决方案**

个人加密资产安全

关于个人加密资产安全的建议指南，欢迎阅读《区块链黑暗森林自救手册》
https://darkhandbook.io

线上热资产安全解决方案

简介

线上热资产主要是指加密货币私钥放置在线上服务器中对应的资产，需要频繁使用来进行签名交易等，如交易所的热、温钱包等都属于线上热资产。这类资产由于放置在线上服务器中，被黑客攻击的可能性大大增加，是需要重点防护的资产。由于私钥的重要性，提高安全存储等级（如硬件加密芯片保护）、去除单点风险等都是防范攻击的重要手段。本文推荐以下两个方向来提升线上热资产的安全性。

协同存管方案

此方案旨在解决线上私钥的单点存储及使用的风险。在以前的方案中，解决私钥单点问题主要通过使用多签，而随着区块链的快速发展，链的种类越来越多，传统的多签（如比特币的多签及以太坊的智能合约多签等）无法适用于所有链的多签方式，导致为了不同的链需要开发不同的多签方案，安全流程极其繁琐和不可控；尤其是线上的热资产本身的场景就需要适应多种链及币种，如交易所、量化等场景。

能兼容所有区块链及币种的通用多签方案是最好的方式，而目前最成熟的解决方案是 MPC（安全多方计算）。

关于 MPC 的解释

安全多方计算（英文：Secure Multi-Party Computation）的研究主要是针对无可信第三方的情况下，如何安全地计算一个约定函数的问题。安全多方计算是电子投票、门限签名以及网上拍卖等诸多应用得以实施的密码学基础。

安全多方计算起源于 1982 年姚期智的百万富翁问题，后来 Oded Goldreich 有比较细致系统的论述。

当前 MPC 对区块链的支持需要在算法层面做很多的研究，对科研及算法底层的要求极高，所以推荐使用开源的程序及已有的商业解决方案，主要推荐如下:

通用多签解决方案

安全鹭（Safeheron）

安全鹭（Safeheron）是领先的数字资产安全存管解决方案提供商。安全鹭（Safeheron）以绝对领先的加密技术， 在确保客户完全掌握自己数字资产的前提下， 提供安全高效的存管服务及解决方案。

官网：https://www.safeheron.com

FireBlocks

Fireblocks is an all-in-one platform to store, transfer, and issue digital assets across your entire ecosystem. Fireblocks 是全球较早的提供基于 MPC 技术提供加密货币协同存管方案的服务商。

官网：https://www.fireblocks.com

TSS 开源库

This is an implementation of multi-party {t,n}-threshold ECDSA (Elliptic Curve Digital Signature Algorithm) based on Gennaro and Goldfeder CCS 2018 1 and EdDSA (Edwards-curve Digital Signature Algorithm) following a similar approach. TSS 是币安开源的门限签名的库，目前支持的算法有 ECDSA 及 EdDSA，开发语言为 Go。

开源仓库地址：https://github.com/binance-chain/tss-lib

ZenGo-X multi-party-ecdsa 开源库

Rust implementation of {t,n}-threshold ECDSA (elliptic curve digital signature algorithm). multi-party-ecdsa 是 MPC 钱包方 ZenGo 提供的开源门限签名的库，支持 ECDSA，开发语言为 Rust。

开源仓库地址：https://github.com/ZenGo-X/multi-party-ecdsa

私钥/助记词安全配置方案

在无法使用 MPC 方案的情况下，如一些小型的加密货币服务、或项目已经成熟变更周期长时，可以针对已有的私钥、助记词等存储及使用进行加强处理，目前的安全建议如下：

KMS

KMS（Key Management Service），即私钥管理服务。当下有很多云服务平台提供相关的 KMS 产品，主要的安全建议是要尽可能使用硬件级方案来对私钥进行管理，尽可能确保私钥在服务器上、数据库里、内存中等地方不会明文暴露，并且使用上有完整的流程及日志记录。

硬件方面上，可以使用 HSM 或 TEE 来提升 KMS 服务的安全级别：

HSM

HSM（Hardware security module），即硬件安全模块，是一种用于保障和管理强认证系统所使用的数字密钥，并同时提供相关密码学操作的计算机硬件设备。

目前各大云服务平台都有提供，可以在自己使用的云服务平台上进行搜索了解。

TEE

TEE（Trusted Execution Environment），即可信执行环境，是中央处理器中安全的区域，可以保证其中的程式和资料在机密性和完整性上得到保护。TEE 是隔离的执行环境，可以有安全的机能，例如隔离执行、和 TEE 一起执行的应用程序完整性，也包括其资产的机密性。用一般的术语来说，TEE 提供安全性更高的执行空间，给可信软件执行，其安全性比操作系统（OS）更强，机能性比安全元件（secure element）更多。

目前芯片厂商及云服务平台都有提供，如微软云及 AWS 等。

Hashicorp Vault

Manage Secrets and Protect Sensitive Data

Secure, store and tightly control access to tokens, passwords, certificates, encryption keys for protecting secrets and other sensitive data using a UI, CLI, or HTTP API. Hashicorp Vault 是一种安全管理机密信息的工具，包括开源软件及商业服务。它可以使用 Shamir 分享算法来进行分

片管理，通过多方参与才能启动服务，是比较好的杜绝单点隐患的方式。同时商业版本也可以支持 HSM。

官网：https://www.vaultproject.io　　开源仓库地址：https://github.com/hashicorp/vault

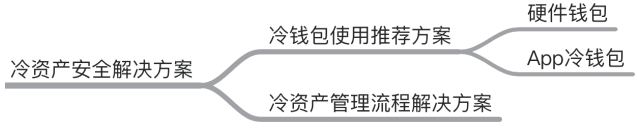
参考文献

安全多方计算	https://zh.wikipedia.org/wiki/安全多方计算
Secure multi-party computation (MPC) 介绍	https://zhuanlan.zhihu.com/p/100648606
HSM 硬件安全模块	https://zh.wikipedia.org/wiki/硬件安全模块
可信执行环境	https://zh.wikipedia.org/zh-cn/可信执行环境

冷资产安全解决方案

简介

加密世界的冷资产主要是指不会经常进行交易的大额资产，并且私钥保存在断网隔离的状态下。理论上来讲冷资产越“冷”越好，即私钥保证永不触网，并且尽量少的交易，尽量避免暴露地址信息等。安全上的方案建议一方面是私钥存储的安全性，做到尽可能的“冷”；另一方面是使用上的管理流程，尽可能避免私钥泄漏、不在预期内的转账或其它未知的行为。



冷钱包使用推荐方案

当前冷钱包大体上分为硬件钱包和App冷钱包两类。

硬件钱包

顾名思义，即使用单独的硬件来存储私钥，通过蓝牙或有线与App、网页等连接来实现签名数据的传输。市面上常见的硬件钱包有以下几种推荐：

Ledger

Ledger 是行业知名硬件钱包，为加密资产提供高级别的安全性，产品结合了安全元件和专为保护资产而设计的专有操作系统。

官网：https://www.ledger.com

Trezor

Trezor 也是区块链行业比较知名的硬件钱包，代码开源。一个亮点是其推荐的 Shamir 备份方案：https://trezor.io/shamir 并且 Trezor Model T 已经支持这种安全的备份模式，值得尝试。

官网：https://trezor.io

imKey

imKey 是一款内置CC EAL 6+ 安全芯片 · 超薄机身 · 蓝牙连接 · 与 imToken 深度集成的硬件钱包，目前支持 BTC、ETH、COSMOS 、EOS 等 imToken 2.0 支持的所有数字资产，未来将会通过升级支持更多的数字资产。

官网：https://imkey.im

Keystone

Keystone 是一款全开源的 Air-Gap 硬件钱包，为了进一步增强安全性，Keystone3 Pro 配备了来自三套不同厂商安全芯片（Secure Elements），并支持在一台设备上贮存多套助记词。

是目前唯一一款与 MetaMask（插件与移动端）均进行了整合的硬件钱包。

官网：https://keyst.one

OneKey

OneKey 是一款有安全芯片且全开源的硬件钱包品牌，其拥有支持手机端、浏览器插件、桌面客户端的完整生态，已经支持 40+ 公链且原生支持几乎所有 EVM 链

官网：https://onekey.so

App 冷钱包

imToken 冷钱包

imToken 冷钱包是指手机在断网的情况下使用 imToken，并且进行离线签名的场景，详细使用方式见以下文章：
如何设置 imToken 冷钱包？（https://support.token.im/hc/zh-cn/articles/360003147833）

冷资产管理流程解决方案

由于冷资产的价值相对较大，属于黑客重点攻击的对象，金钱的诱惑也容易催生出内部作案的可能。如果是一个公司或组织的共有资产，推荐使用一套完善的使用流程来规避被攻击和单点做恶的风险，并且做好流程中每一步的日志记录，以备安全审查。

完善的审批流程

可以在公司内部使用一套成熟的财务审批系统，或者自研的系统，从转账发起到最后转账执行形成流程，需要多方参与审批后才能执行签名转账或其它行为。

多方监督的使用流程

如将存储私钥的硬件钱包放置在保险柜中，在审批完成后，由财务在监督者监督的情形下使用硬件钱包进行转账。

资产交易监控

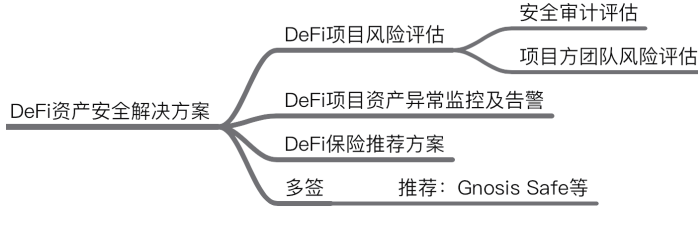
对冷资产的交易进行监控，任何人可以随时提出异议，监控方式可参考：

资产异常监控及追踪解决方案 （https://github.com/slowmist/cryptocurrency-security/blob/-main/Asset-Abnormal-Monitoring-And-Tracking-Solution.md）

DeFi 资产安全解决方案

简介

当前大多数区块链参与方更多的是参与DeFi项目，如挖矿、借贷及理财等。而参与 DeFi 项目本质上是把手中的资产转移或授权给 DeFi 项目方，存在个人极大程度上不可控的安全风险。本方案旨在列出 DeFi 项目的风险点，并且整理出规避这些风险的方式，总体上可分为以下几个方面。



【前】DeFi项目风险评估

识别：

1. 项目方团队背景；
2. 项目方历史项目的安全性；
3. 项目方掌握的权限。

评估：

1. 项目是否有第三方知名审计机构审计；
2. 项目的运行时间是否足够长（如运行3个月以上未发生安全事故的）；
3. 项目的代码质量是否优质；
4. 项目代码是否是透明开源的；
5. 项目方的权限管理是否公开透明；
6. 项目管理的资金规模是否较大；
7. 项目方是否获得知名 VCs（风险投资机构） 投资；
8. 项目社区用户是否活跃，参与的用户数量是否够多。

处置：

1. 仅参与经过(推荐多家)知名安全团队审计过的项目；
2. 仅参与项目代码已开源的项目；
3. 尽量选择参与运行时间长的项目；
4. 尽量选择参与由社区治理的项目；
5. 尽量选择参与项目已经获得多家风险投资机构投资的项目；
6. 为已经参与的项目购买保险。

【中】DeFi 项目资产异常监控及告警

项目异常的情报捕获：

1. 媒体快讯（国内外的平台，如：推特）；
2. 社区消息（如：慢雾区 https://slowmist.io ）；
3. 预警平台告警 （如：RugDoc）；
4. 项目的资产发生异动。

应对措施：

及时取出资产。

【中】多签合约的使用

钱包使用风险：

在使用钱包参与DeFi项目的时候，由于需要频繁使用钱包与DeFi进行交互，在这样的场景下可能由于各种问题导致私钥泄露，有资产被盗的风险。

使用硬件钱包参与DeFi项目又会因为使用的频率较高，与硬件钱包的交互会太繁琐，所以为了同时满足安全与便捷，建议使用多签合约参与DeFi项目。

多签合约推荐：

多签合约可以使用 gnosis-safe 的多签方案快速生成一个多签钱包(如：2/3 或 更多方参与)，避免由于单一钱包的私钥泄露导致的资产被盗。

网址：https://gnosis-safe.io/

【后】DeFi 保险推荐方案

可以根据已参加的 DeFi 项目挑选合适的知名保险项目投保，保险项目参考如下链接：

https://debank.com/projects?tag=insurance

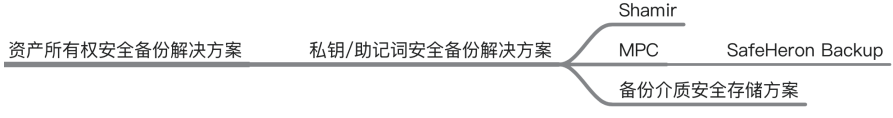
资产所有权安全备份解决方案

简介

加密资产所有权备份，即对私钥或助记词的备份，因为它们承载着对加密货币的完整所有权，一旦被盗或丢失则会损失所有资产。

对于加密资产领域来说，私钥/助记词的备份反而是很大的短板，资产使用上的场景都可以有大量对应的产品来解决，无论是热钱包还是冷钱包，使用上可以保证安全，但很容易忽略其备份的重要性。目前大多数的盗币或者丢币的情况，反而是因为私钥/助记词的备份泄漏或丢失而导致的。备份的重要性等同于加密资产本身，必须要重视起来。

私钥/助记词的备份上也可以考虑降低单点风险，并且使用一些安全的备份方式、介质或流程等。以下是推荐的加密资产所有权备份方案。



Shamir

Shamir 秘密共享机制将秘密拆分后分发给一群人。在这个算法中，原始秘密以适当的方式被拆分成 n 部分，并将拆分后的每一部分分发给不同的参与者。算法的配置决定了最终需要多少个参与者协作才能恢复原始秘密。当使用 Shamir 秘密共享机制保护大容量数据时，一般将对称密钥拆分并进行分布式存储，而非直接针对数据运用 Shamir 算法，这是因为待拆分的秘密容量必须小于秘密共享算法所使用的某些数据。这个算法的 UNIX/Linux 版本被称为 ssss，其他操作系统或编程语言也有相似的应用和开发库。

以上内容节选自《零信任网络：在不可信网络中构建安全系统》

当前比较优秀的基于 Shamir 进行私钥备份分享的有：

Trezor SHAMIR BACKUP

This new security standard, Shamir Backup, counteracts the two greatest risks involved with protecting your recovery seed: theft and destruction.

详细介绍：https://trezor.io/shamir

配合 Trezor 硬件钱包使用是一个不错的安全备份方式。

通用私钥备份及还原程序 by SlowMist (to be done)

慢雾团队基于 Shamir 算法结合自身的安全经验研发了一套私钥分片及备份恢复的程序，整理后未来会进行开源，敬请期待。

MPC

使用 MPC（安全多方计算）可以在最初的私钥生成时即可分为多个分片，将不同的分片分发给一群人，需要恢复时使用特定的程序来还原出原始私钥即可，也是一种优秀的解决单点备份问题的机制。

SafeHeron Backup (to be done)

安全鹭（SafeHeron）提供了一套基于 MPC 的备份还原程序，即将开源。

备份介质安全存储方案

无论是抄写的助记词，存储的 keystore 文件，还是已经分隔后的私钥分片，也需要安全的存储，需要考虑持久性及安全性，保证完全的断网及物理隔离，或完善的加密存储机制。

介质安全

推荐使用金属材质的介质来保存抄写的助记词，或者保证手抄的纸张等结实耐用，如 imKey 的“金钢版助记词密盒”： 采用 304 不锈钢材质，防水、防火、耐腐蚀，支持 2 套助记词保管。更多详细介绍见 imKey 官网：https://imkey.im

存储环境安全

将备份的内容介质存放在安全可靠的环境中，如高规格的保险箱/安全屋等，同时确保其隐蔽性。

离线加密存储

如果存储在电脑或其它电子设备中，需要加密存储，如使用 GPG 等加密工具进行加密，并且将加解密使用的密钥分开存放在其它位置，而不是同一台电脑或设备中。

同时需要考虑断网离线，降低被远程攻击的可能性。

资产异常监控及追踪解决方案

简介

在做好加密资产安全存管系列措施后，为了应对诸如“黑天鹅”之类的意外情况，也需要对相关钱包地址进行监控及异常告警，让每一笔资产转移都能被内部团队确认、验证。

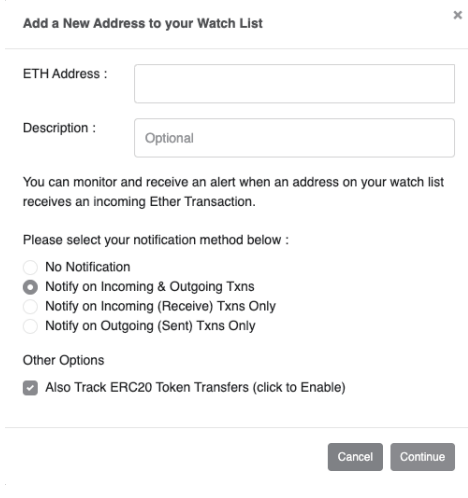
这篇文章我们将展开讲下资产异常监控及追踪的解决方案，主要有以下三部分：

- **异常监控及告警**
- **链上追踪**
- **链下追踪**

异常监控及告警

要对区块链上的钱包地址余额或交易进行监控，可以通过搭建公链全节点的方式，也可以使用第三方提供的数据订阅服务。

监控的逻辑可以根据余额变动来监控，也可以是按交易粒度监控，对目标钱包地址的每一笔链上交易发送提醒。



个人需求可以通过区块浏览器（如 Etherscan）来监控钱包地址，同时满足监控及定制告警的功能（如上图）。除此之外，也可以借助支持导入“观察钱包”的钱包 App（不需要导入私钥助记词，只填写钱包地址），通过 App 的通知功能来实现监控告警功能。

团队需求建议寻找技术工程师，搭建基于消息队列的监控系统及定制化的告警系统（如邮件、Slack、企业微信等消息推送通道），这样系统的稳定性及可用性更有保障。

推荐的监控系统

MistEye

慢雾团队基于多年链上审计及分析经验沉淀出来的链上监控系统，包括恶意/攻击交易识别、合约监控、转账监控、市场价格监控等，详情请看慢雾官网介绍。

链上追踪

如果监控发现钱包地址出现“未授权”的资金转出，可能是遭遇了盗币攻击。可以初步通过区块浏览器确认资金转移路径，对资金留存地址进行监控告警，如果发现转移到中心化交易所，可以联系对应的交易所，提供相关钱包地址、交易 hash 查询。

此外，也可以联系慢雾科技寻求帮助，慢雾 MistTrack 链上追踪服务累计服务 90+ 客户，累计追回资产超 10 亿美金。

依托慢雾 BTI 系统和 AML 系统中 3 亿多地址标签，全面覆盖了全球主流交易所，当被盗资金流入交易所时，慢雾 MistTrack 系统将自动发送交易 hash 和交易所地址等信息给受害者。同时，慢雾 MistTrack 团队将汇总输出被盗资金转移完整链路表、余额停留地址以及洗币情况总结等信息，输出完整、全面的分析报告给受害者。

被盗案件立案成功后，慢雾 MistTrack 团队将协助警方联系被盗资金进入的可调证交易所，对涉案的交易所账号进行调证冻结。

慢雾 MistTrack 链上追踪服务介绍：**https://aml.slowmist.com/cn/case-tracking.html**

链下追踪

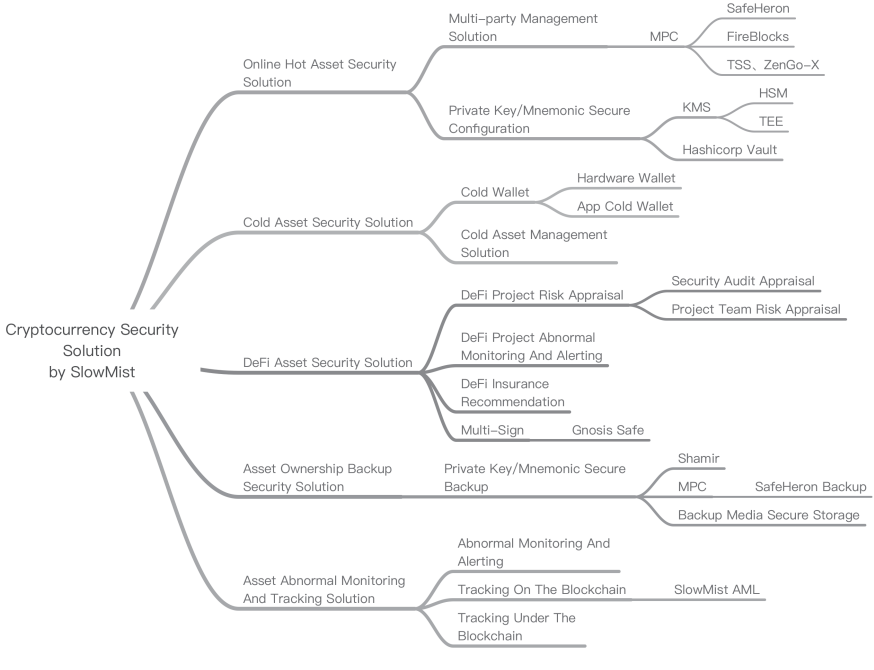
链下追踪主要指的是与区块链无关的信息，例如邮箱账号、IP地址、设备指纹等信息，根据链上追踪得到这些线索后，可以进一步利用相关平台对链下的信息进行分析取证，寻找一切与盗币者有关的信息。

链下追踪主要是使用“社会工程学”与搜索引擎技术，在互联网海量数据中寻找有关联的信息。

Cryptocurrency Security Solution by SlowMist

English Version

This solution is based on the SlowMist security team's many years of practical experience in first-line service to Party A., aims to provide a full range of asset security solutions for participants in the crypto world. We divide the security of crypto assets into the following five major parts, and make a detailed interpretation of each part, including various risks and related solutions.



- Online Hot Asset Security Solution
- Cold Asset Security Solution
- DeFi Asset Security Solution
- Asset Ownership Backup Security Solution
- Asset Abnormal Monitoring And Tracking Solution

Personal Cryptocurrency Security

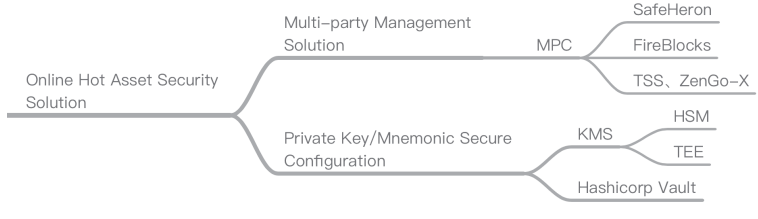
If you want to know personal cryptocurrency security, please read 《Blockchain dark forest selfguard handbook》

https://darkhandbook.io

Online Hot Asset Security Solution

Summary

Online hot assets mainly refer to the assets corresponding to the cryptocurrency private keys placed in online servers, which need to be used frequently for signature transactions, etc., such as hot wallets and warm wallets of exchanges are all online hot assets. Because these assets are placed on online servers, the possibility of being attacked by hackers is greatly increased, and they are assets that require key protection. Due to the importance of private keys, improving the level of secure storage (such as hardware encryption chip protection) and removing single points of risk are important means to prevent attacks. This article recommends the following two directions to improve the security of online thermal assets.



Multi-party Management Solution

This solution aims to solve the risk of single-point storage and use of online private keys. In the previous solution, the single-point problem of private keys was mainly solved by using multi-signature. With the rapid development of blockchain, there are more and more types of chains. Traditional multi-signature (such as Bitcoin's multi-signature and Ethereum smart contract multi-signature, etc.) cannot be applied to the multi-signature method of all chains, which leads to the need to develop different multi-signature schemes for different chains. The security process is extremely cumbersome and uncontrollable; especially the online hot assets themselves need to be adapt to multiple chains and currencies, such as exchanges, quantification and other scenarios.

A universal multi-signature solution compatible with all blockchains and currencies is the best way, and the most mature solution currently is MPC (Secure Multi-Party Computation).

What is MPC

The research of secure multi-party computing is mainly aimed at the problem of how to safely calculate an agreed function without a trusted third party. Secure multi-party computing is the cryptographic basis for the implementation of many applications such as electronic voting, threshold signatures, and online auctions.

Secure multi-party computing originated from Yao Qizhi's millionaire problem in 1982, and later Oded Goldreich made a more detailed and systematic discussion.

At present, MPC's support for blockchain requires a lot of research at the algorithm level, which has extremely high requirements for scientific research and the underlying algorithm. Therefore, it is recommended to use open source programs and existing commercial solutions. The main recommendations are as follows:

Universal multi-signature solution

SafeHeron

SafeHeron is a leading provider of digital asset safe custody solutions. With absolute leading encryption technology, SafeHeron provides safe and efficient depository services and solutions on the premise of ensuring that customers have complete control of their digital assets.

Official website: https://www.safeheron.com

FireBlocks

FireBlocks is an all-in-one platform to store, transfer, and issue digital assets across your entire ecosystem. FireBlocks is one of the earliest service providers in the world to provide collaborative cryptocurrency depository solutions based on MPC technology.

Official website: https://www.fireblocks.com

TSS open source library

This is an implementation of multi-party {t,n}-threshold ECDSA (Elliptic Curve Digital Signature Algorithm) based on Gennaro and Goldfeder CCS 2018 1 and EdDSA (Edwards-curve Digital Signature Algorithm) following a similar approach.

GitHub repository: https://github.com/binance-chain/tss-lib

ZenGo-X multi-party-ecdsa open source library

Rust implementation of {t,n}-threshold ECDSA (elliptic curve digital signature algorithm).

GitHub repository: https://github.com/ZenGo-X/multi-party-ecdsa

Private Key/Mnemonic Secure Configuration

In the case that the MPC solution cannot be used, such as some small cryptocurrency services, or when the project has matured and the change cycle is long, the storage and use of existing private keys, mnemonics, etc. can be strengthened. The current security recommendations are as follows :

KMS

KMS (Key Management Service), that is, private key management service. Currently, there are many cloud service platforms that provide related KMS products. The main security recommendation is to use hardware-level solutions to manage private keys as much as possible, and ensure that private keys are not exposed in plaintext on the server, database, memory, etc. , And there is a complete process and log record for use.

In terms of hardware, HSM or TEE can be used to improve the security level of KMS services:

HSM

HSM (Hardware security module), namely the hardware security module, is a computer hardware device used to protect and manage the digital key used by the strong authentication system and provide related cryptographic operations at the same time.

At present, all major cloud service platforms are provided, and you can search on the cloud service platform you use.

TEE

TEE (Trusted Execution Environment) is a secure area in the central processing unit, which can ensure that the programs and data in it are protected in terms of confidentiality and integrity. TEE is an isolated execution environment, which can have secure functions, such as isolated execution, the integrity of applications executed with TEE, and the confidentiality of its assets. In general terms, TEE provides a more secure execution space for trusted software to execute. Its security is stronger than the operating system (OS), and its functionality is more than secure elements.

Currently, chip manufacturers and cloud service platforms provide such services, such as Microsoft Cloud and AWS.

Hashicorp Vault

Manage Secrets and Protect Sensitive Data

Hashicorp Vault is a tool for securely managing confidential information, including open source software and commercial services. It can use the Shamir sharing algorithm to manage the shards, and the service can be started only through the participation of multiple parties, which is a better way to eliminate single-point hidden dangers. At the same time, the commercial version can also support HSM.

Official website: https://www.vaultproject.io
GitHub repository: https://github.com/hashicorp/vault

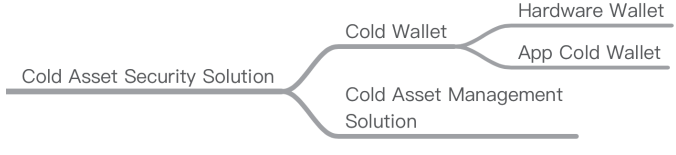
References

安全多方计算	https://zh.wikipedia.org/wiki/安全多方计算
Secure multi-party computation (MPC) 介绍	https://zhuanlan.zhihu.com/p/100648606
HSM 硬件安全模块	https://zh.wikipedia.org/wiki/硬件安全模块
可信执行环境	https://zh.wikipedia.org/zh-cn/可信执行环境

Cold Asset Security Solution

Summary

The cold assets in the crypto world mainly refer to large assets that are not frequently traded, and the private keys are kept in a disconnected state. In theory, the colder the asset is, the "cold" the better, that is, the private key is guaranteed to never touch the Internet, and there are as few transactions as possible to avoid exposing address information. On the one hand, the security proposal is to store the private key as "cold" as possible; on the other hand, it is to use the management process to avoid private key leakage, unexpected transfers or other unknown behaviors as much as possible.



Cold Wallet

The current cold wallets are roughly divided into hardware wallets and App cold wallets.

Hardware Wallet

As the name implies, a separate hardware is used to store the private key, and the signature data is transmitted through Bluetooth or wired connection with App, web pages, etc. The common hardware wallets on the market have the following recommendations:

Ledger

Ledger is a well-known hardware wallet in the industry that provides a high level of security for encrypted assets. The product combines secure components and a proprietary operating system designed to protect assets.

Official website: https://www.ledger.com

Trezor

Trezor is also a well-known hardware wallet in the blockchain industry with open source code. One highlight is its recommended Shamir backup solution: **https://trezor.io/shamir**, And Trezor Model T already supports this safe backup mode, which is worth trying.

Official website: https://trezor.io

imKey

imKey is a hardware wallet with built-in CC EAL 6+ security chip, ultra-thin body, Bluetooth connection, and deep integration with imToken. It currently supports all digital assets supported by imToken 2.0 such as BTC, ETH, COSMOS, EOS, etc. It will be upgraded in the future to support more digital assets.

Official website: https://imkey.im

Keystone

Keystone is an open-source airgap hardware wallet that utilizes an embedded system. The device is equipped with three secure element chips. One of Keystone's unique features is to support multiple recovery seed phrases.

Keystone is currently the only hardware wallet that is compatible with both MetaMask desktop and mobile versions.

Official website: https://keyst.one

OneKey

OneKey is a fully open-source hardware wallet brand equipped with a security chip. It features a comprehensive ecosystem that includes support for mobile devices, browser extensions, and desktop clients. The wallet currently supports over 40 public blockchains and natively supports almost all EVM chains.

Official Website: https://onekey.so

App Cold Wallet

imToken Cold Wallet

The imToken cold wallet refers to the scenario where the mobile phone uses imToken and performs offline signatures when the mobile phone is disconnected from the network. For detailed usage, please refer to the following article: How do I use cold wallet (formerly: watch wallet)? (https://support.token.im/hc/en-us/articles/360003147833)

Cold Asset Management Solution

Since the value of cold assets is relatively large, they are the key targets of hackers, and the temptation of money can easily give rise to the possibility of internal crimes. If it is a common asset of a company or organization, it is recommended to use a complete set of usage procedures to avoid the risk of being attacked and single-point evil, and to make a log record of each step in the process for security review.

Perfect approval process

A mature financial approval system or a self-developed system can be used within the company to form a process from the initiation of the transfer to the final execution of the transfer. It requires

multiple parties to participate in the approval before the signature transfer or other actions can be performed.

Multi-party supervised use process

For example, the hardware wallet that stores the private key is placed in a safe, and after the approval is completed, the finance will use the hardware wallet to transfer assets under the supervision of the supervisor.

Asset transaction monitoring

To monitor the transaction of cold assets, anyone can raise an objection at any time. The monitoring method can refer to:Asset Abnormal Monitoring And Tracking Solution (https://github.com/slowmist/cryptocurrency-security/blob/-main/en/Asset-Abnormal-Monitoring-And-Tracking-Solution.md)

DeFi Asset Security Solution

Summary

At present, most blockchain participants are more involved in DeFi projects, such as mining, lending and financial management. Participating in a DeFi project is essentially the transfer or authorization of assets in the hand to the DeFi project party, and there is a security risk that is largely uncontrollable by individuals. This solution aims to list the risk points of the DeFi project and sort out the ways to avoid these risks. Generally speaking, it can be divided into the following aspects.

[Previous] DeFi Project Risk Assessment

Identify:

1. Project team background;
2. The safety of the historical project of the project party;
3. The authority controlled by the project party.

Evaluate:

1. Whether the project is audited by a well-known third-party audit team;
2. Whether the operation time of the project is long enough (for example, no safety accident occurred during operation for more than 3 months);
3. Whether the code quality of the project is high quality;
4. Whether the project code is transparent and open source;
5. Whether the authority management of the project party is open and transparent;
6. Whether the scale of project management assets is large;
7. Whether the project party has obtained investment from well-known VCs (venture investment institutions);
8. Whether the project community users are active and whether the number of participating users is large enough.

Handle:

1. Only participate in projects that have been audited by (recommended multiple) well-known security teams;
2. Only participate in projects whose project code has been open source;
3. Try to choose projects that have a long running time;
4. Try to choose to participate in projects governed by the community;
5. Try to choose projects that have been invested by multiple venture capital institutions to participate in the project;
6. Purchase insurance for the projects you have participated in.

[Participating] DeFi project asset abnormal monitoring and alerting

Intelligence capture of project anomalies:

1. Media news (platforms at home and abroad, such as Twitter);
2. Community news (eg: SlowMist Zone https://slowmist.io);
3. Early warning platform alarm (such as RugDoc);
4. The assets of the project are changed.

Handle:

Take out assets in time.

[Participating] Use of multi-signature smart contracts

Risks of using wallet:

When using a wallet to participate in a DeFi project, due to the need to frequently use the wallet to interact with DeFi, in such a scenario, the private key may be leaked due to various problems, and there is a risk of asset theft.

Using hardware wallets to participate in DeFi projects will be too cumbersome to interact with hardware wallets because of the high frequency of use. Therefore, in order to meet safety and convenience at the same time, it is recommended to use multi-signature smart contracts to participate in DeFi projects.

Multi-sign smart contract recommendation:

Multi-signature contracts can use gnosis-safe's multi-signature solution to quickly generate a multi-signature wallet (for example, 2/3 or more parties participate) to avoid asset theft due to the leakage of the private key of a single wallet.
Official website：https://gnosis-safe.io

[After] DeFi Insurance Recommended Plan

You can select suitable well-known insurance projects to insure according to the DeFi projects you have participated in. For insurance projects, refer to the following link: https://debank.com/projects?tag=insurance

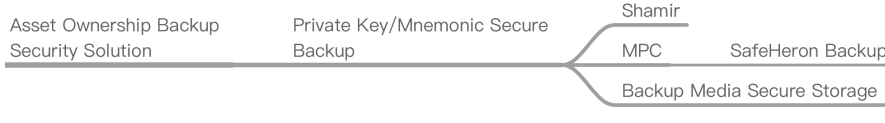
Asset Ownership Backup Security Solution

Summary

Asset ownership backup in the crypto world, that is, the backup of private keys or mnemonic, because they carry the complete ownership of the encrypted currency, once they are stolen or lost, all assets will be lost.

For the field of crypto assets, the backup of private keys/mnemonic is a big shortcoming. Asset usage scenarios can be solved by a large number of corresponding products. Whether it is a hot wallet or a cold wallet, the use can be guaranteed safe, but it is easy to overlook the importance of its backup. At present, most of the theft or loss of coins is caused by the leakage or loss of the backup of the private key/mnemonic. The importance of backup is the same as the crypto asset itself, and it must be taken seriously.

For the backup of private keys/mnemonics, you can also consider reducing single points of risk, and use some secure backup methods, media, or processes. The following is a recommended backup solution for the ownership of crypto assets.



Shamir

The Shamir secret sharing mechanism splits the secret and distributes it to a group of people. In this algorithm, the original secret is split into n parts in an appropriate way, and each part after splitting is distributed to different participants. The configuration of the algorithm determines how many participants are ultimately required to cooperate to recover the original secret. When using

the Shamir secret sharing mechanism to protect large-capacity data, the symmetric key is generally split and stored in a distributed manner, instead of directly applying the Shamir algorithm to the data. This is because the capacity of the secret to be split must be smaller than that used by the secret sharing algorithm some of the data. The UNIX/Linux version of this algorithm is called ssss, and other operating systems or programming languages have similar applications and development libraries.

The above content is excerpted from "Zero Trust Network: Building a Security System in an Untrusted Network"

The current excellent private key backup sharing based on Shamir are:

Trezor SHAMIR BACKUP

This new security standard, Shamir Backup, counteracts the two greatest risks involved with protecting your recovery seed: theft and destruction.

Detailed introduction: **https://trezor.io/shamir**
Use with Trezor hardware wallet is a good safe backup method.

Universal private key backup and restore program
by SlowMist (to be done)

The SlowMist team has developed a set of private key slicing and backup and recovery procedures based on the Shamir algorithm and its own security experience. After finishing, it will be open sourced in the future.

MPC

Using MPC (Secure Multi-Party Computing) can be divided into multiple shards when the initial private key is generated, and different shards can be distributed to a group of people. When recovery is needed, use a specific program to restore the original private key. It is also an excellent mechanism to solve the single point backup problem.

SafeHeron Backup (to be done)

SafeHeron provides a set of MPC-based backup and restore programs, which will be open source soon.

Backup media secure storage solution

Whether it is copied mnemonics, stored keystore files, or separated private key fragments, secure storage is also required. Persistence and security need to be considered to ensure complete disconnection and physical isolation, or perfect encryption storage mechanism.

Media security

It is recommended to use metal media to store the copied mnemonic, or to ensure that the handwritten paper is strong and durable, such as imKey's "Golden Steel Version Mnemonic Secret Box": It is made of 304 stainless steel, waterproof, fireproof, and corrosion-resistant, and supports 2 sets of mnemonic storage. For more detailed introduction, please refer to imKey official website: **https://imkey.im**

Storage environment security

Store the backup content media in a safe and reliable environment, such as a high-standard strongbox/safe house, etc., while ensuring its concealment.

Offline encrypted storage

If it is stored in a computer or other electronic device, it needs to be stored encrypted, such as using encryption tools such as GPG for encryption, and the keys used for encryption and decryption are stored separately in other locations, rather than in the same computer or device.

At the same time, you need to consider disconnection and offline to reduce the possibility of remote attacks.

Asset Abnormal Monitoring and Tracking Solution

Summary

After implementing a series of measures for the safe custody of encrypted assets, in order to deal with unexpected situations such as "black swan", it is also necessary to monitor the relevant wallet addresses and abnormal alarms, so that every asset transfer can be confirmed by the internal team. verify.

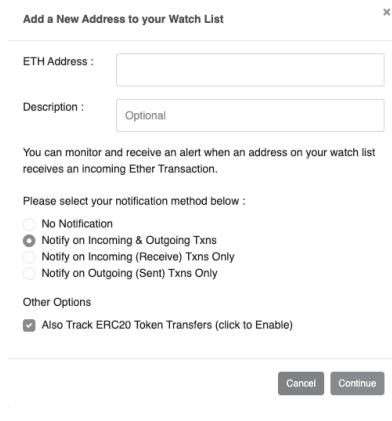
In this article, we will start to talk about the solution of asset abnormality monitoring and tracking, which mainly has the following three parts:

- **Abnormal Monitoring And Alerting**
- **Tracking On The Blockchain**
- **Tracking Under The Blockchain**

Abnormal Monitoring And Alerting

To monitor the wallet address balance or transaction on the blockchain, you can build a full node on the public chain, or you can use a data subscription service provided by a third party.

The monitoring logic can be monitored according to balance changes, or according to transaction granularity, sending reminders for each on-chain transaction of the target wallet address.



Personal needs can monitor the wallet address through a block explorer (such as Etherscan), and at the same time meet the monitoring and customized alarm functions (as shown above). In addition, you can also use the wallet App that supports the import of "watch wallet" (no need to import the private key mnemonic, just fill in the wallet address), through the notification function of the App to realize the monitoring alarm function.

The team's needs are recommended to find technical engineers to build a monitoring system based on message queues and a customized alarm system (such as mail, Slack, corporate WeChat and other message push channels), so that the stability and availability of the system are more guaranteed.

Recommended Monitoring System

MistEye

The on-chain monitoring system developed by the SlowMist team based on years of on-chain auditing and analysis experience includes malicious/attack transaction identification, contract monitoring, transfer monitoring, market price monitoring, etc. For details, please see the SlowMist official website.

Tracking On The Blockchain

If the monitoring finds "unauthorized" fund transfers from the wallet address, it may be due to a currency theft attack. You can preliminarily confirm the fund transfer path through the blockchain browser, and monitor and alert the fund retention address. If you find that it is transferred to a centralized exchange, you can contact the corresponding exchange to provide related wallet addresses and transaction hash queries.

In addition, you can also contact SlowMist Technology for help. SlowMist's MistTrack on-chain tracking service has served 90+ customers and recovered more than 1 billion US dollars in assets.

Relying on more than 300 million address tags in the SlowMist BTI system and AML system, it fully covers mainstream exchanges around the world. When the stolen assets flow into the exchange, the SlowMist MistTrack system will automatically send information such as the transaction hash and exchange address to the victim. At the same time, the SlowMist MistTrack team will summarize and output the complete link list of stolen assets transfer, the balance staying address, and the summary of coin laundering, and output a complete and comprehensive analysis report to the victim.

After the successful filing of the theft case, the SlowMist MistTrack team will assist the police in contacting the adjustable stock exchange where the stolen assets entered, and freeze the account of the exchange involved.

SlowMist MistTrack on-chain tracking service introduction:
https://aml.slowmist.com/case-tracking.html

Tracking Under The Blockchain

Off-chain tracking mainly refers to information that has nothing to do with the blockchain, such as email account numbers, IP addresses, device fingerprints and other information. After these clues are obtained based on the on-chain tracking, the relevant platforms can be further used to analyze and collect evidence on off-chain information. Look for all the information related to the thief.

Off-chain tracking mainly uses "social engineering" and search engine technology to find relevant information in the massive data of the Internet.

https://www.slowmist.com

SlowMist is a blockchain security firm established in January 2018. The firm was started by a team with over ten years of network security experience to become a global force. Our goal is to make the blockchain ecosystem as secure as possible for everyone. We are now a renowned international blockchain security firm that has worked on various well-known projects such as HashKey Pro, OSL, MEEX, BGE, BTCBOX, Bitget, BHEX.SG, OKX, Binance, HTX, Amber Group, Crypto.com, etc.

SlowMist offers a variety of services that include but are not limited to security audits, threat information, defense deployment, security consultants, and other security-related services. We also offer AML (Anti-money laundering) software, Vulpush (Vulnerability monitoring) , SlowMist Hacked (Crypto hack archives), FireWall.x (Smart contract firewall) , Safe Staking and other SaaS products. We have partnerships with domestic and international firms such as Akamai, BitDefender, FireEye, RC^2, TianJi Partners, IPIP, etc.

By delivering a comprehensive security solution customized to individual projects, we can identify risks and prevent them from occurring. Our team was able to find and publish several high-risk blockchain security flaws. By doing so, we could spread awareness and raise the security standards in the blockchain ecosystem.

Services and Products

Blockchain Security Audit

Provide security services for CEX, DEX, DeFi, GameFi, NFT, Wallets, Blockchains

Red Teaming

Evaluate personal, organizational, supply chain, office, and physical security risks

Security Monitoring

MistEye, the meticulously developed system that provides comprehensive dynamic security monitoring services for Web3 projects

MistTrack Service

Digital assets were unfortunately stolen, MistTrack provides a glimmer of hope. Customers served: 90 + ; Cumulative recovered assets: $1,000,000,000 +

Security Consulting

Provide technical, risk management and emergency response support, along with recommendations for improvements

Blockchain AML

An AML/CFT compliance solution that employs on-chain analytics to trace illicit funds

SLOWMIST AML

https://aml.slowmist.com

Promoting the compliance, security, and healthy development of the Web3 industry

Shine a Beam of Light in the Dark Forest of Blockchain

SlowMist AML Solutions

Compliance

- Meeting regulatory requirements
- Address risk identification
- Threat intelligence network
- Build an anti-money laundering alliance

Investigation

- Emergency response
- Hacker profile
- Intercept stolen funds
- Recovering stolen funds

Audit

- HKSFC's 23 Compliance Requirements
- OWASP's 27 Compliance Requirements
- Over 170 security audit items compiled by SlowMist

Products

MistTrack KYT

Combined with the industry-leading blockchain intelligent analysis, easy-to-use interface, and real-time API.

MistTrack Investigation

An investigation platform that provides wallet address analysis, fund monitoring, and fund tracking.

Malicious Address Library

It covers relevant content from the dark web to hundreds of global trading platforms, providing comprehensive intelligence support for tracking hacker attacks and money laundering behavior.

Services

Case Evaluation

If your cryptocurrency has been stolen, the community provides free case evaluation assistance services.

Case Tracking

We provide professional tracking services, organize the transfer links of the stolen funds, provide verifiable information, and assist the police in solving the case.

Case Report

We perform a one-time comprehensive analysis of the chain asset theft case and issue a report on the case tracking.

MISTTRACK

https://misttrack.io

A Crypto Tracking and Compliance Platform for Everyone

MistTrack is an anti-money laundering tracking system developed by the SlowMist AML. We use on-chain analytics to assist in the tracing of illicit funds.

Free 30-Day Trial

1K + Address Entities

500K + Threat Intelligence Addresses

300M + Addresses Labeled

90M + Risky Addresses Identified

Wallet Address Risk Assessment

Determine if an address is managed by crypto exchanges, sanctioned entities, darknet markets, or mixers, as well as whether the wallet address was engaged in illicit activities.

Investigate and Track Crypto Activity

Tracking and identifying the movement of crypto assets on wallet addresses as well as monitoring and notifying transfers in real-time. Combining on-chain and off-chain data into a single panel to provide significant technological support for legal investigations.

User Portrait Analysis

Collect all linked address labels via one or more wallet addresses and characterize its profile based on ENS name, transaction action, transaction time, and entities involved.

Follow the Whales & KOLs

Add whales and KOLs (Key Opinion Leaders) to your private favorites list to receive insight into their on-chain activity. You can monitor their transactions in real-time to give you a better understanding of their actions.

Focusing on Blockchain Ecosystem Security

Website

https://slowmist.com
https://aml.slowmist.com
https://misttrack.io

GitHub

https://github.com/slowmist

X (Twitter)

https://x.com/SlowMist_Team
https://x.com/MistTrack_io

Email

team@slowmist.com