如何审计一份智能合约?

- Author: xxxeyJ@Phaseless

大多审计师(Auditor)或者安全研究员(Security Researcher)都有一套根据过往实战经验总结出来的智能合约审计思路，但是到目前为止，国内安全圈还没有公开讨论过类似的思路。那么我这篇小作文就简单来写一下我是如何自拿到一个项目开始对 Smart Contract 进行安全审计的。

其实无论是哪次审计，审计的目标是谁，周期有多长，复杂性如何。你都需要做出一些固定的前期准备，比如编辑器(VSCode)，区块链浏览器(E/B/Pxxx-Scan)，安全工具(S/E/S/S/O/M/M)、辅助工具/插件(SVD/Surya/etc)一系列工具。（其实说白了，只要前期在本地配置好这些相关工具，后期几乎无需做出变更）

在这其中，我最喜欢的一款工具是 **VSCode** 中的 `SVD` 插件，它集成了多款辅助工具，可以用它查看智能合约的继承树，方法签名，Graph，UML图表，以及其它合约相关信息，这可以帮助我快速理清各个智能合约之间的联系

侦查

《**阅读文档**》，一般来讲，项目方将业务逻辑编写成智能合约代码之后，为方便工程师一类的用户了解其中功能用途，都会发布一份查阅文档以供其加深业务理解。

对于审计师而言，首先要做的第一步应该是通读文档全文，通过书面化的形式将各个业务逻辑关系梳理清晰。加深项目运作原理的理解，并检查它的代码复用性怎么样，有没有使用到外部库的扩展，有无拷贝其它类似项目的代码。

而后将智能合约代码库导入至编辑器中，在结合之前通过文档了解到的业务逻辑与代码实例相关联后理清智能合约的层次结构和功能，并使用一些辅助工具加深理解。

分析

工具审计

使用诸如 (S/E/S/S/O/M/M) 一类的安全工具以自动化的形式对智能合约进行安全分析，这里需要了解的是，工具只能根据过往的漏洞且具备一定规则的情况下对合约进行分析，而如今的脆弱点早已不是前几年那样整型溢出/重入攻击 (个例通过组合拳) 满天飞的状态，这些基础漏洞类型几乎已经不复存在。但即便如此，使用安全工具进行审计也是为了确认代码原型没有较为明显的基础漏洞。

人工审计

人工审计相对工具审计的区别在于人工会根据自身过往经验对代码做出判断，审计师水平参差不齐也导致了审计出的结果存在差异。

通过前期对文档以及代码的关联性进行全局分析，利用一些工具，比如(Surya)将不同的 **Function Visibility** 进行分类，优先检查可见性为 **Public** 、**External** 的方法，寻找其中调用了哪些 **Internal**、**Private** 方法，业务功能需求是否与代码实现功能相匹配，逻辑关系是否安全，是否会修改系统状态(状态变量)，检查状态的修改对于应用场景的影响是怎样的。

倘若审计的项目 Fork 自其它仓库，可以检查项目方是否在原有的基础进行过二开。因为本身项目方的水平也参差不齐，可能有些片段原本没有问题，但是在项目方根据自身需求添加代码后反而出现了问题。这时，只要关注这些片段即可。

尾语

上述提到的这些安全工具都可以在我的Github项目找到原型。

https://github.com/xxxeyJ/Awesome-Blockchain-Security#tools