استخراج درخت نحو و قوانین انتزاع برای تحلیل آسیب پذیری کدهای solidity

دانشجو: محدثه رفیعی

استاد راهنما: دکتر فاطمه قاسمی

دانشکده مهندسی برق و کامپیوتر، دانشگاه تهران

مقدمه / خلاصه

کدهای سالیدیتی کدهایی هستند که به طور خاص برای نوشتن قراردادهای هوشمند در پلتفرم‌های بلاک چین طراحی شده است. قراردادهای هوشمند توافق‌نامه‌های خود اجرایی هستند که شرایط توافق به صورت مستقیم در کد نوشته شده‌اند. این قراردادها برروی شبکه‌های بلاکچین اجرا می‌شوند و قوانین خود را بدون نیاز به واسطه‌گرها به صورت خودکار اجرا می‌کنند.

این کدها با کارهایی مانند مدیریت تراکنش‌های توکن، پیاده‌سازی مکانیزم‌های حاکمیت یا مدیریت قوانین یک پروتکل مالی غیرمتمرکز سروکار دارند.

تبدیل این کدها به زبان‌های مدرن‌تر به ما این امکان را خواهد داد که بتوانیم این کدها را بهینه کنیم و بر اساس قوانین بتوانیم آن‌ها را در برابر حملات مقاوم سازیم.

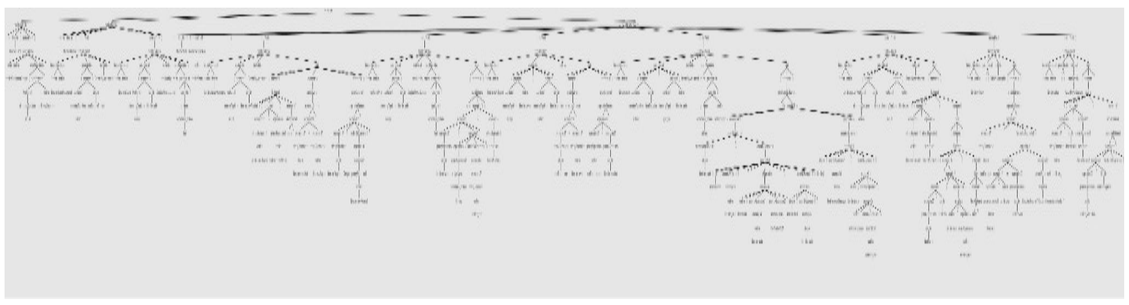
این پروژه به این موضوع خواهد پرداخت که:

۱- چگونه از این کدها درخت نحو استخراج شده است.

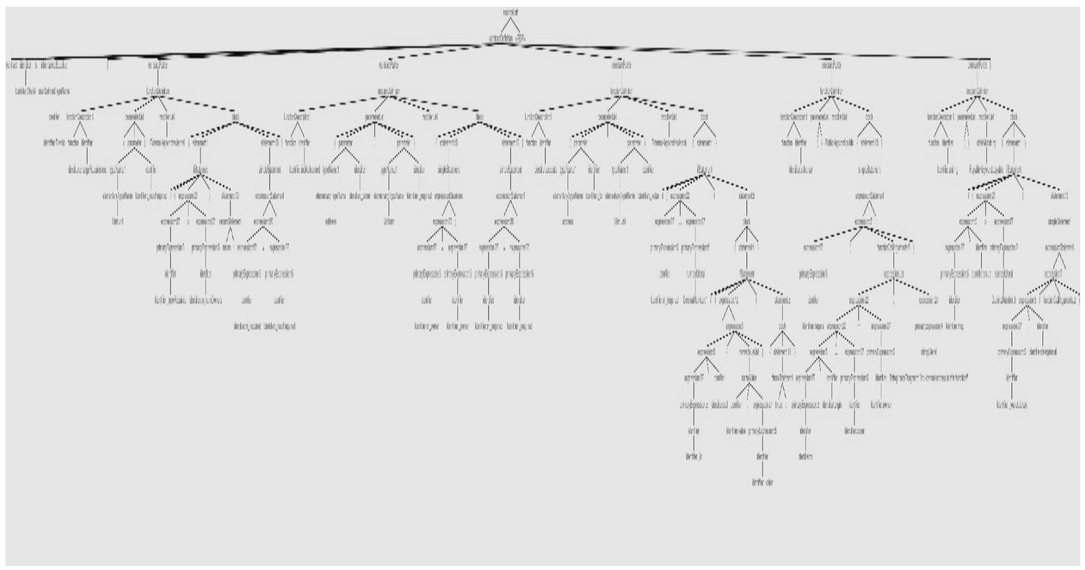
۲- چه قوانین حذفی روی درخت نحو اعمال شده است که بررسی کدهای غیر مهم در سناریوهای حمله اتفاق نیفتد.

روش/ساختار/مدل پیشنهادی

در این قسمت تلاش شده است با استخراج درخت نحو انتزاعی از کدهای سالیدیتی ابتدا قوانین حذف و بستار روی درخت ایجاد شده اعمال کنیم تا به صورت ساده تر و حجم کد کمتر بتوانیم آن را به زبان های سطح بالاتر مانند mcrl2 تبدیل کنیم و همچنین بتوانیم آسیب‌پذیری این کدها را به صورت کارآمد بررسی کنیم.



شکل بالا درخت نحو انتزاعی قبل از اعمال قوانین حذف و بستار است شکل زیر بعد حجم کد ساخته شده بعد از اعمال قوانین حذف را نشان خواهد داد.



نتایج

در این پروژه توانستیم با قوانین حذف و بستار و ایجاد AST زبان بتوانیم از مجموعه حالات قابل بررسی به طور ویژه بکاهیم و بتوانیم بعد از استخراج درخت نحو به راحتی این درخت را به هر زبان سطح بالاتری برای بررسی نقاط ضعف امنیت برای مثال mcrl2 تبدیل نمود.

```solidity
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.13;

contract DAO {
    mapping (address => uint) public userBalances;

    function addToBalance() public payable {
        userBalances[msg.sender] += msg.value * (10**18);
    }

    function getUserBalance(address user) public view returns(uint) {
        return userBalances[user];
    }

    function withdrawBalance(uint amount) public {
        if(userBalances[msg.sender] >= amount){
            (bool temp,) = msg.sender.call{value : (amount / (10**18))}("");
            require(temp, 'error');
            userBalances[msg.sender] -= amount;
        }
    }
}
```

۱) قبل از قوانین حذف و بستار

```solidity
contract DAO{
    mapping(address=>uint)public userBalances;
    function addToBalance()public payable{
        userBalances[msg.sender]+=msg.value*(10**18);
    }
    function withdrawBalance(uint amount)public{
        if(userBalances[msg.sender]>=amount){
            (booltemp,)=("");
            require(temp,'error');
            userBalances[msg.sender]-=amount;
        }
    }
}
```

۲) بعد از اعمال قوانین

جمع بندی

در این تحقیق در گام نخست ما بر ادبیات موضوع گذری کردیم و متوجه شدیم که مفهوم قراردادهای هوشمند در زمینه بلاک‌چین و سالیدیتی چیست. متوجه شدیم قراردادهای هوشمند تکه کدهایی هستند که به صورت خود ران اجرا می‌شوند و کارهایی را بر اساس قوانین خاصی انجام می‌دهند.
متوجه شدیم به دلیل ماهیت این قراردادها و بلاک‌چین و این کدها مستعد حملات امنیتی هستند. برای جلوگیری از این حملات باید بتوانیم این کدها را ساده‌تر کنیم و از ابزارها برای تشخیص تعدادی از حملات استفاده کنیم. در فصل دوم در مفاهیم عمیق‌تر شده و نمونه‌ای از درخت نحو را دیدیم.

در فصل سوم دیدیم چه روش‌هایی برای انجام این عمل موجود است و برای اینکه بتوانیم به راحتی کدهای سالیدیتی را به کدهای زبان‌های سطح بالاتر تبدیل کنیم از درخت نحو استفاده نموده‌ایم; و همچنین تعدادی نمونه از این کدها مشاهده کردیم.

در فصل چهارم روند پیاده‌سازی و قوانین حذف و بستار توضیح داده شد این قوانین برای ساده‌سازی استفاده شده اند تا تعداد مجموعه حالات بررسی نقاط ضعف امنیتی به راحتی صورت بگیرد.

مراجع اصلی

"Introduction to Static Analysis Using Solidity" by Philippe Charland