

## Design Model for Extensible Architecture of Smart Contract Vulnerability Detection Tool

Yun-seok Choi<sup>1</sup>, Wan Yeon Lee<sup>2</sup>

<sup>1, 2</sup> Professor, Department of Computer Science, Dongduk Women's University, Korea

<sup>1</sup>cooling@dongduk.ac.kr, <sup>2</sup>wanlee@dongduk.ac.kr

### Abstract

Smart contract, one of the applications of blockchain, is expected to be used in various industries. However, there is risks of damages caused by attacks on vulnerabilities in smart contract codes. Tool support is essential to detect vulnerabilities, and as new vulnerabilities emerge and smart contract implementation languages increase, the tools must have extensibility for them. We propose a design model for extensible architecture of smart contract vulnerability detection tools that detect vulnerabilities in smart contract source codes. The proposed model is composed of design pattern-based structures that provides extensibility to easily support extension of detecting modules for new vulnerabilities and other implementation languages of smart contract. In the model, detecting modules are composed of independent module, so modifying or adding of module do not affect other modules and the system structure.

**Keywords:** Blockchain, Smart contracts, Ethereum, Vulnerability, Software Architecture, Extensibility

### 1. Introduction

Blockchain, introduced for peer-to-peer payments of Bitcoin in 2009, is expand its application field to new services such as smart contracts[1,2]. Ethereum is a platform that implements blockchain-based smart contracts and can be used for developing a wide range of applications such as e-commerce, production and manufacturing, banking, etc[3,4]. As applications of smart contract have expanded to various fields, the vulnerability of smart contracts has emerged. In June 2016, Ether was stolen due to Reentrancy attack, one of the vulnerabilities of smart contract code, resulting in a loss of about 60 million USD, and in November 2017, an accident of freezing Ether worth about 300 million USD Occurred. It is considered that smart contracts vulnerabilities and the attacks on them will increase more and more[5,6]. Therefore, various studies are being conducted to identify vulnerabilities in smart contract and to develop vulnerability detection tools. For well-known smart contract vulnerabilities, classification and related test cases and detailed information can be viewed on online[7-9], various tools such as Oyente, Mythril, and Smartcheck are used for vulnerability analysis[2,10,11]. The software architecture of the detection tool should be extensible in order to support new vulnerabilities and the variety programming languages of smart contracts. We propose a design model of



extensible architecture for smart contract vulnerability detection tools. The proposed model is composed of each of vulnerability detection as an independent module and applying extensible design patterns. This paper describes the model of the extensible architecture for vulnerability detection tool which targets are smart contract source codes before publishing. Vulnerability detection techniques and algorithms are out of scope. The rest of this paper organized as follows. In Section 2, we review the backgrounds of our work, design model of extensible architecture is shown in Section 3. A case study and discussion about the design model are shown in Section 4. Section 5 concludes the paper.

## 2. Backgrounds

### 2.1 Smart Contract and Vulnerability

Blockchain which is a basis technology of cryptocurrency, consists of a sequence of timestamped blocks that can store transaction records like a general ledger in the real world[1,12]. Blockchain has advantages of decentralization, security, anonymity, auditability of transactions and so on. So, various studies are being conducted to utilize blockchain. In addition to cryptocurrency, the most representative example of using blockchain is smart contracts, and platforms such as Ethereum, Cardano, and NEO are being used to implement smart contracts[3]. Smart contracts that utilize the advantages of blockchain can guarantee the integrity of transactions in a distributed environment and reduce the risk of manipulation because all data is shared with users. Therefore, it is expected to be used for the development of distributed applications in various fields such as finance, copyright, logistics, electronic voting, continuous history information management, and so on. With the increasing applications of smart contract, security problems due to errors and vulnerabilities in smart contract implementation codes are emerging[5,6,13]. As a result, various studies are being conducted to analyze and supplement vulnerability of smart contract implementation code. There are more than 30 vulnerabilities in the SWC Registry, which summarizes the smart contract vulnerabilities[7], and as the applications of smart contract increase in the future, the types of vulnerability may increase. Table 1 shows the examples of smart contract vulnerability.

**Table 1. Examples of smart contract vulnerability**

Title	SWC Code	Description
Improper Adherence to Coding Standards	SWC-100/108	Unauthorized or unintended state changes can be made
Integer Overflow and Underflow	SWC-101	An overflow/underflow can be occurred
Floating Pragma	SWC-103	An outdated compiler version can be affected the contract system negatively
Improper Access Control	SWC-105/106	Malicious parties can self-destruct the contract
Re-entrancy	SWC-107	Different invocations of a target function to interact in undesirable ways

These vulnerabilities similar to logical errors of general applications, so they should be able to be identified and supplemented before smart contract publishing.

### 2.3 Extensibility Related to Architecture of Vulnerability Detection Tools

An architecture of vulnerability detection tool needs to be configured to have the following characteristics



of extensibility. First, it should be easy to extend detecting functionality for new types of vulnerability. It is necessary to be able to extend new functionality and to supplement the existing functionality without modifying the structure of the tool. Similar types of functionality should be systematized and easily extended. Second, the architecture of the tool should be support smart contract source codes implemented in a new implementation language. In the case of Ethereum, which uses solidity normally, it is being extend kinds of the smart contract implementation languages such as Python, Java, and Go. Therefore, it is necessary to be able to extend functionality to recognize source codes of smart contracts written in a new implementation language.

### 3. Design Model of Extensible Architecture

In this section, we will show a design model of extensible architecture for vulnerability detection tool. The proposed model is targeted at an architecture of a tool which scans smart contract codes before publishing. The model is composed of a structure which can extend detection functionality by an independent module.

#### 3.1 Design Model Structure

The proposed model is designed to represent specific or multiple vulnerabilities. Each vulnerability is composed of an independent module, and vulnerability detection is performed using a set of vulnerability modules. The VulnerabilityType which represents a detection module is composed of the strategy pattern[14]. Each instance of the VulnerabilityType has a common interface for detection and can be selected dynamically to use. The attribute matchingPattern which expressed by regular expressions can be used to detect vulnerability. The VulnerabilityTypes can be categorized by types of vulnerability and programming languages of smart contract source codes. The AbstractDetectorFactory creates concrete VulnerabilityTypes for each type of vulnerability and smart contract implementation language. It is composed of the factory method pattern[14], so a new type of VulnerabilityType can be added without modifying of the structure. Since the strategy pattern and the factory method pattern are applied, the DetectorClient can perform detections with the same usage regardless the type of VunerabilityType. Figure 1 shows the design model structure of extensible architecture.

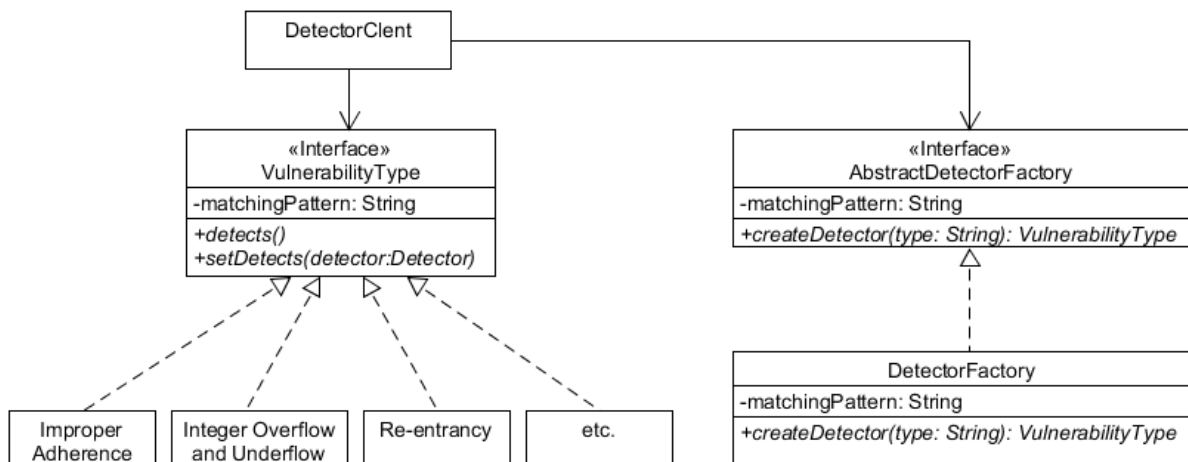
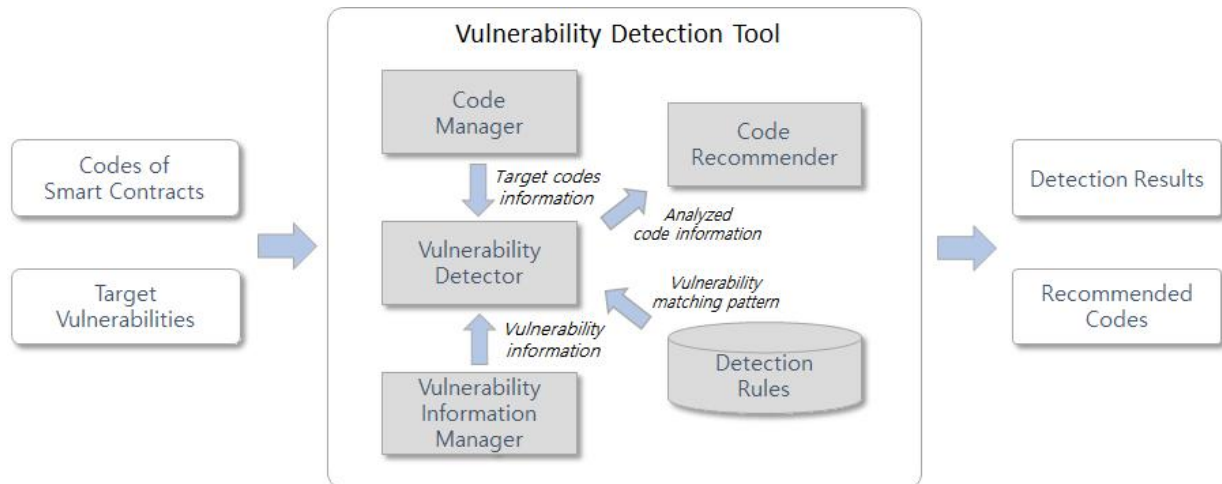


Figure 1. Design model structure



### 3.2 System Architecture

A system architecture used the proposed design model selects the detecting rules for each programming language according to the source code of the smart contract, which is the target of vulnerability detection, and dynamically selects the detection module according to the type of vulnerability to be detected. Figure 2 shows an example of system architecture of the tool based on the design model.



**Figure 2. System architecture**

The Code Manager receives the smart contract source codes as input, identifies the implementation language, and stores codes to a structure that can easily perform detection. The identification results of the Code Manager are used to select detection modules in the Vulnerability Detector which is implemented based on the proposed design model. The vulnerability detecting function is provided by the Vulnerability Information Manager, and the pattern matching rule for vulnerability detection is obtained from the Detection Rules. The Vulnerability Detector performs vulnerability detection for smart contract source codes structured by the Code manager. The analysis results of each vulnerability are sent to the Code Recommender, and the Code Recommender generates recommendations that can be used to complement detection results.

### 4. A Case study and Discussion

In order to verify the usefulness of the proposed model, a smart contract vulnerability detection tool based on the model and system architecture was implemented with Java. Each of detecting module is the type of VulnerabilityType in the design model, it can be modified or added without modifying the structure of the tool, because it has the common type and interface. Vulnerability detecting techniques and algorithms are out of scope this study, so we used well known detecting techniques and algorithms to implement each of the VulnerabilityType modules. The tool based on the proposed model consists of independent module by the type of vulnerability, so it can be able to extend the detecting module according to type of the vulnerability or the implementation language without changes of the tool structure. Figure 3 shows parts of implementations of the proposed model. In Figure 3, (a) shows the parent class of vulnerability detecting classes which is the type of the VulnerabilityType and (b) shows a package structure of detection modules. If a new vulnerability detecting module is needed, implement a class inheriting from the VulnerabilityType and add it to this package structure. (c) shows a part of an implemented class for detecting a vulnerability. Since all detecting modules implement the common interface detects(), the modules can have the same usage regardless of the type of



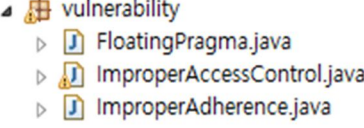
vulnerability when detecting.

```
public abstract class VulnerabilityType implements ScanString{

    private VulnerabilityType vType;
    private String matchingPattern;

    abstract public List<ScanResult> detects (String sourceCode, List<String> sourceCodeByLine);

    public void setDetector(VulnerabilityType vType) {}
}
(a)
```


 (b)

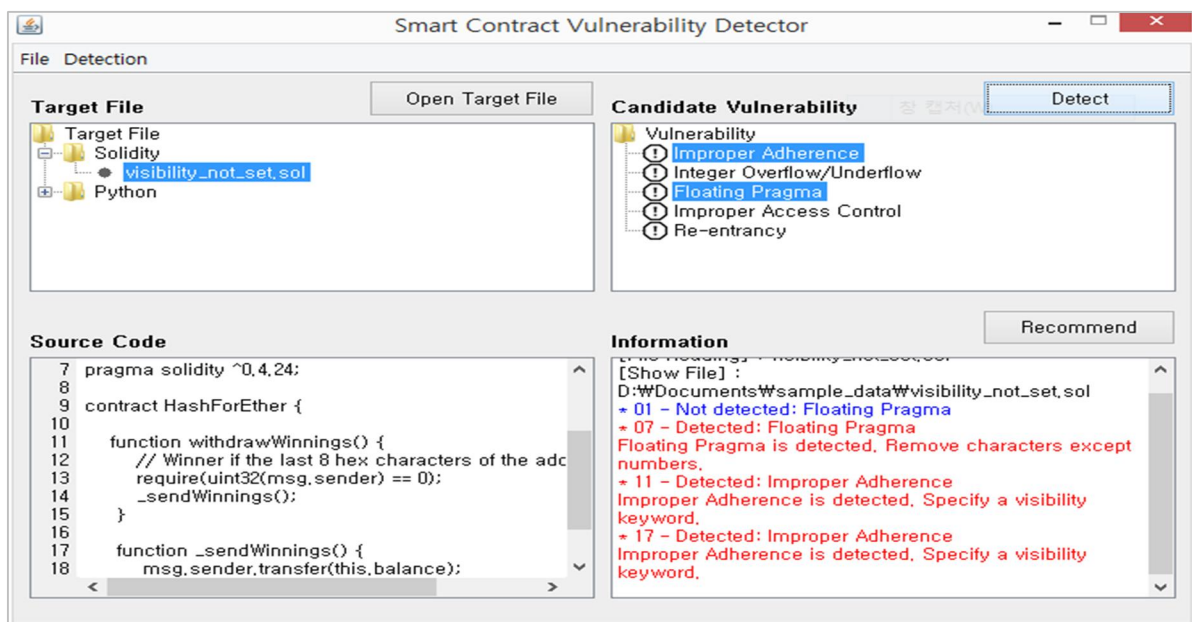
```
public class FloatingPragma extends VulnerabilityType implements Commons {

    @Override
    public List<ScanResult> detects(String sourceCode,
        List<String> sourceCodeByLine) {
    }
}
(c)
```

**Figure 3. Implementation of the design model**  
**(a) VulnerabilityType class, (b) Package structure, (c) An implemented class**

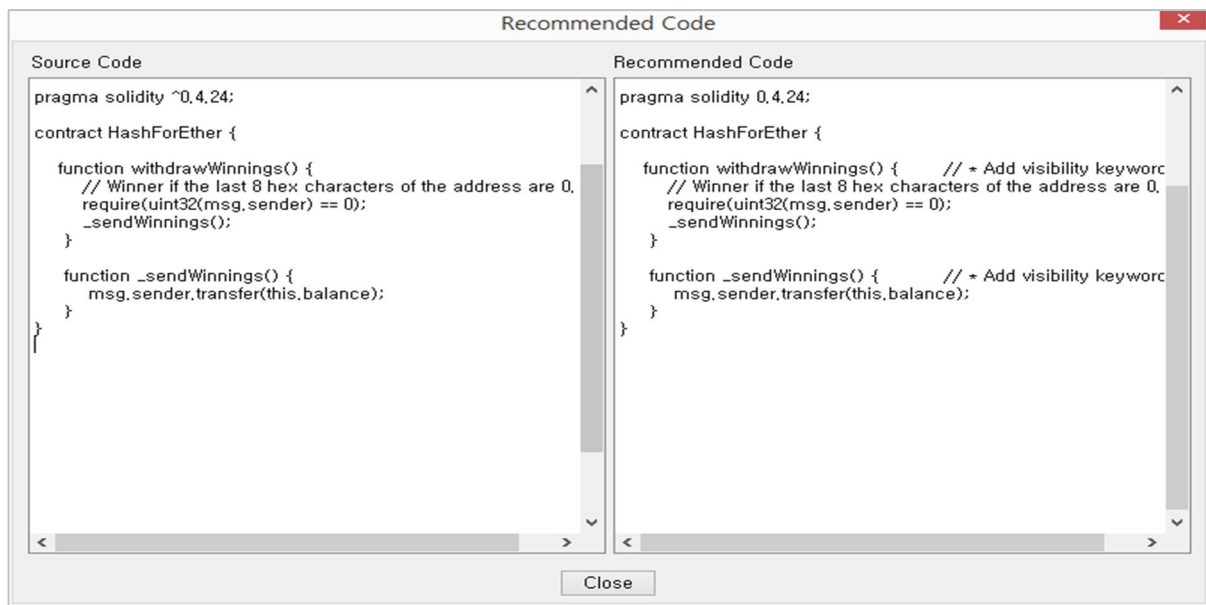
The target of the implemented tool can be smart contract source codes implemented by solidity or python. Multiple vulnerabilities can be checked when it selected on the candidate vulnerability list. If a vulnerability of smart contract source code is detected, information about the vulnerability is displayed, and recommended code information supplementing the vulnerability is provided.

Figure 4 shows the execution of the vulnerability detection tool and Figure 5 shows the recommended code. As a result of checking the exploitable vulnerability against the visibility\_not\_set.sol source code, a test case of the improper adherence in [7], In addition to the improper adherence, the intended vulnerability in the test cases, the results of the floating pragma vulnerability were detected. In the recommend code, we were able to confirm the recommendations to modify the floating pragma of the original source code and to supplement the improper adherence. In Figure 5, the first line of code was changed to prevent the floating pragma and annotations were inserted to inform the improper adherence.



**Figure 4. Smart contract vulnerability detection tool**





**Figure 5. Recommended code**

Through the implementation of the tool, we verified that the tool implemented based on the proposed model can be added a new vulnerability detecting module without modifying the existing structure. As the design model intended, detection modules can be added for which new type of vulnerability or smart contract source codes using another implementation language.

## 5. Conclusion

Smart contracts are expanding applications to various fields based on the advantages of the blockchain, but there is risks of attack due to the vulnerability of smart contract codes. As new vulnerabilities emerge and smart contract implementation languages increase, vulnerability detection tools must have extensibility for them. In this paper, we proposed a design model for extensible architecture of smart contract vulnerability detection tool. The proposed model was composed of design pattern-based structures that provides extensibility to easily support the extension of modules for new vulnerabilities and other implementation languages. A detecting functionality was composed of independent module, so modifying or adding of module do not affect other modules and the system structure. The tool implemented based on the proposed model showed that it can be extended vulnerability modules without modifying the existing structure. Therefore, applying the proposed model can be expected to facilitate the extension for detection functionality.

It is necessary to study models which can publishing smart contract codes and detecting deployment errors on multiple environments. In addition, it is necessary to study the modeling for vulnerability detection based on byte codes of smart contracts.

## Acknowledgement

This research was supported by the Dongduk Women's University Grant, 2019

## References

- [1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. <https://bitcoin.org>



- [2] L. Luu, D. Chu, H. Olickel, P. Saxena, and A. Hobor, "Making Smart Contracts Smarter," in Proc. 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 254-269, Oct. 2016.  
DOI: <https://doi.org/10.1145/2976749.2978309>
- [3] Ethereum Foundation, Ethereum Whitepaper, <https://ethereum.org/en/whitepaper/>
- [4] N.F. Samreen and M.H. Alalfi, "Reentrancy Vulnerability Identification in Ethereum Smart Contracts," in Proc. 2020 IEEE International Workshop on Blockchain Oriented Software Engineering, pp. 22-29, Feb.18, 2020.  
DOI: <https://doi.org/10.1109/IWBOSE50093.2020.9050260>
- [5] A. Dika and M. Nowostawski, "Security Vulnerabilities in Ethereum Smart Contracts," in Proc. 2018 IEEE International Conference on Internet of Things and IEEE Green Computing and Communications and IEEE Cyber, Physical and Social Computing and IEEE Smart Data, pp. 955-962, July 2018.  
DOI: [https://doi.org/10.1109/Cybermatics\\_2018.2018.00182](https://doi.org/10.1109/Cybermatics_2018.2018.00182)
- [6] P. Qian, Z. Liu, Q. He, R. Zimmermann, and X. Wang, "Towards Automated Reentrancy Detection for Smart Contracts Based on Sequential Models," IEEE Access, Vol. 8, pp. 19685-19695, Jan. 2020.  
DOI: <https://doi.org/10.1109/ACCESS.2020.2969429>
- [7] SWC Registry(Smart Contract Weakness Classification and Test Cases), <https://swcregistry.io/>
- [8] CVE(Common Vulnerabilities and Exposures), <https://cve.mitre.org/>
- [9] CWE(Common Weakness Enumeration), <https://cwe.mitre.org/>
- [10] B. Mueller, A Framework for Bug Hunting on the Ethereum Blockchain, <https://github.com/ConsenSys/mythril>
- [11] S. Tikhomirov, E. Voskresenskaya, I. Ivanitskiy, R. Takhaviev, E. Marchenko, and Y. Alexandrov, "Smartcheck: Static analysis of Ethereum smart contracts," in Proc. IEEE/ACM 1st Int. Workshop Emerg. Trends Softw. Eng. Blockchain (WETSEB), pp. 9-16, May/Jun. 2018.  
DOI: <https://doi.org/10.1145/3194113.3194115>
- [12] Z. Zheng, S. Xie, H.N. Dai, X. Chen, and H. Wang, "Blockchain challenges and opportunities," International Journal of Web and Grid Services(IJWGS), Vol. 14, No. 4, pp. 352-375, Oct. 2018.  
DOI: <https://doi.org/10.1504/IJWGS.2018.095647>
- [13] W.Y. Lee and Y.S. Choi, "Vulnerability and Cost Analysis of Heterogeneous Smart Contract Programs in Blockchain Systems," Current Trends in Computer Sciences & Applications, Vol. 2, Issue 1, pp. 142-145, Feb. 2020.  
DOI: <https://doi.org/10.32474/CTCSA.2020.02.000126>
- [14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design pattern, Addison Wesley, pp. 107-116, pp.315-324, 1995