KIPS Tr. Software and Data Eng.
Vol.5, No.3 pp.139~144 pISSN: 2287-5905

http://dx.doi.org/10.3745/KTSDE.2016.5.3.139

Cost Based Vulnerability Control Method Using Static Analysis Tool

Ki Hyun Lee† · Seok Mo Kim†† · Young B. Park††† · Je Ho Park††††

ABSTRACT

When, Software is developed, Applying development methods considering security, it is generated the problem of additional cost. These additional costs are caused not consider security in many developing organization. Even though, proceeding the developments, considering security, lack of ways to get the cost of handling the vulnerability throughput within the given cost. In this paper, propose a method for calculating the vulnerability throughput for using a security vulnerability processed cost-effectively. In the proposed method focuses on the implementation phase of the software development phase, leveraging static analysis tools to find security vulnerabilities in CWE TOP25. The found vulnerabilities are define risk, transaction costs, risk costs and defines the processing priority. utilizing the information in the CWE, Calculating a consumed cost in a detected vulnerability processed through a defined priority, and controls the vulnerability throughput in the input cost. When applying the method, it is expected to handle the maximum risk of vulnerability in the input cost.

Keywords : Vulnerability Treatment, Cost Based, Static Analysis

정적 분석 툴을 이용한 비용 기반의 취약점 처리 방안

이 기 현† · 김 석 모†† · 박 용 범††† · 박 제 호††††

요 약

소프트웨어 개발 시 보안을 고려한 개발방법의 적용은 추가비용을 발생시키고, 이러한 추가 비용은 많은 개발조직에서 보안을 고려하지 못하는 원인이 된다. 보안을 고려한 개발을 진행 하더라도, 주어진 비용 내에서 처리할 수 있는 취약점 처리량을 산출하는 방법이 부족한 실정이다. 본 논문에서는 보안 취약점 처리 비용을 효율적으로 사용하여 취약점 처리량을 산출하는 방법을 제안한다. 제안한 방법에서는 소프트웨어 개발단계 중 구현 단계에 중점을 두고, 정적 분석 툴을 활용하여 CWE TOP25에 대한 보안 취약점을 찾아낸다. 찾아진 취약점은 CWE의 정보를 활용하여, 각 취약점의 위험도, 처리 비용, 비용 당 위험도를 정의하고, 처리 우선순위를 정의한다. 정의된 우선순위를 통하여 탐지된 취약점 처리에 소모되는 비용을 산출하고, 투입 비용 내에서 취약점 처리량을 제어한다. 본 방법을 적용하면, 투입 비용 내에서 최대의 취약점의 위험도를 처리할 수 있을 것으로 기대된다.

키워드 : 취약점 처리, 비용 기반, 정적 분석

1. 서 론

요즘 사이버 공격의 75%는 소프트웨어의 취약점을 공격한다는 보고가 되었다[1]. 이에 따라서 소프트웨어의 보안에 대한 중요도가 증가하였다. 하지만 보통의 경우 소프트웨어 개발 시에 보안을 고려하지 않고 개발하고 있으며, 소프트웨어 개발 시에 보안을 고려하지 않고 개발을 할 경우에는, 소프트웨어에서 발생하는 취약점의 처리를 유지보수 단계에서 패치나 업데이트를 통해서 해결하고 있다. 유지보수단계에서 취약점을 처리 하는 것은 개발 단계에서 취약점을 처리하는 것보다 보다 많은 비용을 필요로 하게 된다[2, 3]. 이에 따라서 개발 단계에서부터 보안을 고려한 개발 방법에 대한 움직임이 활성화 되었으며[4, 5], 실제로 보안성을 고려한 개발 방법은 소프트웨어가 가진 취약점을 감소시키는 성과를 내고 있다[5]. 하지만 보안을 고려한 개발 방법을 사용할 경우에는 추가 비용이 들어가는 문제가 발생하게 되며, 추가 비용의 문제는 개발 프로젝트의 실패로 이어질 수 있다. 실제로 개발 프로젝트의 약 52% 정도가 계획된 비용보다 초과하여 프로젝트를 진행하였으며[6], 추가로 보안성을

※ 본 연구는 미래창조과학부 및 한국인터넷진흥원의 "2015년도 고용 계약형 정보보호 석사과정 사업"의 연구 결과로 수행되었음.
※ 이 논문은 2015년도 정부(미래창조과학부)의 재원으로 정보통신기술진흥센터의 지원을 받아 수행된 연구임(No.10044457, 자율지능형 지식/기기 협업 프레임워크 기술 개발).
† 준 회 원 : 단국대학교 컴퓨터학과 소프트웨어보안 석사과정
†† 비 회 원 : 단국대학교 컴퓨터학과 소프트웨어보안 석사과정
††† 종신회원 : 단국대학교 전자계산학과 교수
†††† 종신회원 : 단국대학교 전자계산학과 부교수
Manuscript Received : November 30, 2015
First Revision : January 22, 2016
Accepted : January 22, 2016
* Corresponding Author : Young B. Park(ybpark@dankook.ac.kr)

고려하게 된다면, 기존의 비율보다 더 높은 수치를 가지게 될 것이다. 이러한 이유로 주어진 비용 안에서 보안성을 고려하기 위한 방법이 필요로 하게 되었다. 본 논문에서는 보안 취약점 처리 비용에 대한 보안 취약점의 위험도를 정의하고, 보안 취약점 처리 비용 당 보안 취약점의 위험도의 우선순위를 정함으로써, 정해진 비용 안에서 최대의 보안 취약점을 처리할 수 있는 방법을 제시한다.

2. 관련 연구

2.1 정적 분석

정적 분석이란 프로그램을 실행시키지 않고, 프로그램의 코드를 분석하는 방법으로, 프로그램 코드 상에 존재하는 결함들을 찾아낸다. 정적 분석 방법은 검토(review), 소프트웨어측정(metrics), 데이터흐름 분석, 제어흐름 분석, 정형기법(formal method) 코드 취약점(vulnerability) 분석 등이 있다. 정적 분석에는 보통 툴을 이용하여 분석을 하게 되는데, 툴을 사용할 경우 보다 정확한 정적 분석이 가능하다. 실제로 취약점 분석에 정적 분석이 많이 사용되고 있다[7, 8]. 따라서 본 논문에서는 개발 단계의 코드의 취약점 분석을 위하여 정적 분석 툴을 이용하였다.

2.2 위험도 분석

CWE(Common Weakness Enumeration)[9]는 소프트웨어에서 발생되는 취약점에 대한 정보를 모아 놓은 데이터베이스로써, SANS(SysAdmin, Audit, Network and Security)와 Mitre에 의해서 발표되었다. CWE는 각 항목에 대한 특징과 점수를 제공하는데, CWE는 점수를 통하여, 보안 취약점의 위험도를 평가하게 된다. CWE의 점수는 CWSS(Common Weakness Scoring System)을 이용하여 구하게 되며, CWSS는 취약점의 위험도 점수를 계산할 때, CWSS Metric Group을 이용한다[10].

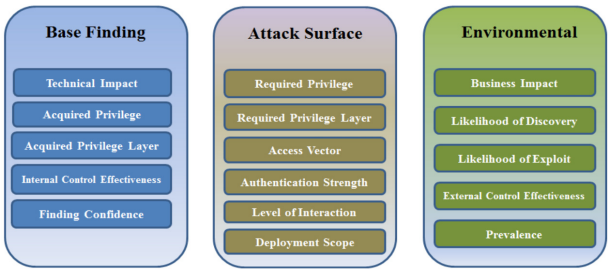

Fig. 1. CWSS Metric Group

Fig. 1은 CWSS Metric Group를 요약한 그림이다. Fig. 1에서와 같이 CWSS Metric Group에서는 취약점에 대한 속성을 정의하고 있다. CWSS에서는 CWSS Metric Group에 정의된 취약점의 속성에 대한 값을 수치화 하고, CWSS에서 제안한 위험도 산출식을 이용하여, 취약점에 대한 위험도를 점수로 표현하게 된다.

2.3 비용 추정

비용 추정이란, 어떠한 작업을 수행하는데 소모되는 비용을 예측하는 것으로, 비용을 추정하는 방법은 추정 대상에 따라서 많은 방법들이 존재한다[11]. 본 논문에서는 여러 추정 방법 중 유사 추정법을 사용하며, 유사 추정법이란, 과거의 유사한 데이터를 통해서 가격을 추출하는 방법으로, 경험을 토대로 비용을 추정하는 방법이다[12]. 실제 경험을 토대로 비용을 추정하기 때문에 보다 객관적인 비용의 추정이 가능하게 된다. 본 논문에서는 CWE의 경험적 데이터를 통하여 취약점 처리 비용의 추정하는데 유사 추정법을 이용한다.

3. 취약점 처리 효율지수 정의

3.1 취약점의 위험도 정의

본 논문에서 목적으로 하는 비용 당 위험도의 효율성을 고려하기 위해서는, 먼저 보안 취약점 위험도의 정의가 선행되어야 한다. 만약에, 보안 취약점 위험도가 정의되어 있지 않는 경우에는, 각 보안 취약점이 소프트웨어에 미치는 영향을 고려하기 힘들기 때문에, 보안 취약점 위험도를 정의한다. 보안 취약점 위험도는 CWE TOP25에서 제공하는 Score를 사용하였다. CWE에서 Score는 CWSS를 사용하여 구하게 되며, CWSS는 CWSS Metric Group을 이용하여 구하게 된다. CWSS의 Metric에는 각 취약점이 가지는 위험속성에 대한 내용들이 정의되어 있으며, 각각은 위험속성에 대한 수치를 가지고 있다. 이러한 CWSS Metric Group 수치를 이용하여 계산된 CWE, Score는 1~100의 값을 가지게 된다. 여기에서 위험도가 높은 취약점 일수록 높은 Score를 가지게 되며, 즉 Score가 높은 취약점은 위험도가 높다는 것을 의미한다.

3.2 취약점의 처리 비용 수치 정의

취약점 처리 비용은 CWE Top25의 Summary에 있는 처리비용 항목을 사용하였다. CWE의 경우에는 처리 비용을 수치가 아닌, High, Medium, Low 등의 등급으로 정의하고 있기 때문에, 처리 효율지수를 산출하기 위하여, CWSS의 정보를 이용하여 각 등급에 대한 수치를 부여하고, 각 등급에 대한 가중치를 부여하였다. 가중치 값으로는 0.1을 사용 하였다. 0.1의 가중치를 부여한 이유는 CWE Top25 Summary와 CWSS Metric Group에 공통적으로 존재하는 Prevalence 항목을 이용하였기 때문이다. Prevalence 항목은 High를 0.9로 기준으로 하여 가중치 값을 0.1을 사용하고 있다. 위 내용을 정리하면 아래의 Table 1과 같다.

Table 1. Definition of Treatment Cost Score

Treatment cost rating	Definition	Treatment cost value
High	architecture change	0.9
Medium	code change in a single file and component	0.8
Low	code change in a single block or function	0.7

이렇게 Table 1에서 정의된 처리 비용의 수치를 CWE Top 25의 Summary에 적용하면, 각 취약점별 처리 비용수치를 구할 수 있으며, 이렇게 구해진 처리 비용 수치는 취약점 처리효율지수를 산출하는데 이용된다.

3.3 취약점 처리 효율지수의 산출

본 논문에서 말하는 취약점 처리 효율지수란 Cost Per Vulnerability, 즉 처리 비용 단위당 취약점의 위험도의 크기로 정의할 수 있다. CWE에서 정의한 취약점의 위험도를 그대로 사용하지 않고, 따로 처리 효율지수를 정의하는 이유는 취약점의 위험도가 크다고 하여, 처리 비용이 높지는 않기 때문이다. 이러한 특성 때문에 처리 비용에 따른 취약점의 위험도를 나타내는 처리 효율지수를 정의하였으며, 처리 효율지수를 구하기 위해서 앞 장에서 정의한 취약점의 위험도와 처리 비용 수치를 사용 한다. 효율성 점수를 구하는 산출 식은 아래와 같다.

처리 효율지수 = 취약점의 위험도 / 처리 비용 수치

산출 식을 통하여 구해진 처리 효율지수는 높은 값을 가질수록 처리 비용을 사용하여 보다 높은 보안 취약점의 위험도를 처리할 수 있다는 것을 의미한다. 이렇게 구해진 처리 효율지수를 통하여 취약점 처리의 우선순위를 결정할 수 있게 된다.

4. 취약점 처리 제어

4.1 정적 분석 툴을 통한 취약점 검출

본 논문에서는 취약점 검출을 위해 정적 분석 툴을 사용한다. 취약점 처리의 제어 방안은 CWE의 정보를 이용하기 때문에, 정적 분석 툴은 CWE에 존재하는 취약점을 검출할 수 있어야 한다. 이러한 취약점을 검출할 수 있는 툴로는, 상용 툴인 Fortify[13]와 오픈 소스인 PMD가 있다. PMD의 경우에는 취약점을 검출할 수 있는 진단 규칙을 별도로 적용해야 한다[14]. 이러한 정적 분석 툴을 사용하면, 코드 상에 존재하는 취약점을 검출할 수 있으며, 검출된 CWE 취약점에 대해서 검출된 취약점 개수와 취약점이 검출된 위치 등에 대한 정보를 얻을 수 있다. 본 논문에서는 취약점 처리 제어를 위해서, 검출된 취약점 항목과 검출된 취약점 개수를 이용하게 된다. 검출된 취약점의 항목정보는 검출된 취약점의 위험도와 처리 비용 수치를 정의하는데 이용되며, 정의된 위험도와 처리 비용 수치를 이용하여, 처리 효율지수를 산출한다. 검출된 취약점의 개수의 경우에는 검출된 취약점의 처리 소모비용을 산출에 이용된다. 이러한 정보를 이용하여 투입된 비용 내에서 처리의 효율을 높이는 방향으로 취약점 처리 방안을 수립한다. 본 논문에서는 CWE Top 25에 존재하는 취약점이 1개씩 검출된 데이터를 이용하였다.

4.2 취약점 처리 투입 비용의 정의

본 논문에서 투입 비용은 프로젝트 전체 비용 중에서 취약점 처리에 투입되는 비용을 말한다. 하지만 비용은 프로젝트의 성격이나 규모에 따라서 투입되는 비용의 절대적인 크기가 전부 다르기 때문에, 취약점 처리에 투입되는 비용을 산출해내는데 어려움이 있다. 이러한 문제를 해결하기 위해서 투입비용의 정의를 전체 비용 중에서 취약점 처리에 투입하는 비용의 비율로 정의한다. 비용을 비율로 정의할 경우 전체 프로젝트의 크기가 다르더라도, 취약점 처리에 투입하는 비용 정도를 판단할 수 있다. 이렇게 정의된 투입 비용의 비율의 상수는 투입 비용의 값으로 사용된다. 예를 들어 프로젝트 A에서 취약점 처리 비용으로 30%를 투자 하였다면, 투입 비용은 30의 값을 가지게 된다.

4.3 취약점 처리 소모비용 산출

취약점 처리에 대해 최적화된 계획을 세우기 위해서는 취약점 처리 시에 소모되는 총비용을 먼저 정의해야 한다. 취약점 처리 시에 소모되는 총 비용은 검출된 각 취약점의 처리 비용의 합으로 구할 수 있다. 이때 합해지는 각 취약점은 우선순위 지표에 따른 순서로 계산되어지며, 검출된 취약점의 수만큼 반복해서 합을 구하게 된다.

1) 취약점 항목별 처리 소모비용 산출

CWE Summary의 처리 비용은 등급으로만 분류가 되어 있기 때문에, 실제로 취약점 처리에 들어가는 비용을 산출하기가 힘들다. 처리 비용 등급에 해당하는 취약점 처리비용 산출하기 위해서, 개발 비용 상수를 이용하여 곱해 주게 되는데, 여기서 프로젝트 상수는 프로젝트에서 구현에 들어가는 비용으로 정의한다. 이렇게 구해진 처리 비용은 취약점을 처리하는데 소모되는 비용을 의미한다.

Table 2. Vulnerability treatment Consumed Cost Formula

Treatment cost rating	Formula
High	(The number of files to be modified / The total number of code files)* Constant development costs
Medium	(1/The total number of code files)* Constant development costs
Low	1/The total number of code function* Constant development costs

정적 분석 툴을 사용하여 검출된 취약점 항목에 대해서, 검출된 취약점 항목의 수정 비용 등급 정보 위의 Table 2를 적용하여, 각 취약점 항목을 처리하는데 소모비용을 산출할 수 있다.

2) 검출된 취약점별 처리 소모비용 산출

취약점 별 처리 소모비용은 검출된 취약점이 가지는 처리 비용에 비례하고, 취약점이 검출된 수에 비례 한다. 아래 Equation (1)은 취약점에 대한 처리 비용과 전체 취약점을 처리하기 위한 수식을 정리한 것이다.

$$VTC_i = f(TC_i, DN_i) * IC \quad (1)$$

Where $f()$ is a monotonic incremental function for (RC_i, DN_i)

VTC_i = 우선순위가 i인 취약점의 처리 소모비용 합
TC_i = 우선순위가 i인 취약점의 처리 소모비용
DN_i = 우선순위가 i인 검출된 취약점 개수
i = 취약점 처리 효율 우선순위

위에서 정의된 Equation (1)의 i는 정수 값을 가지며, 취약점의 처리 효율에 따른 우선순위를 의미한다. 한 항목에 대한 취약점의 처리 소모비용을 계산할 때는 우선순위의 중요도가 낮지만, 총 소모비용을 계산할 때에는 우선순위가 중요한 요소로 작용하게 된다.

3) 검출된 전체 취약점 처리 소모비용 산출

취약점 처리에 소모되는 총 비용은 대상으로 한 소스코드에서 검출된 모든 취약점을 처리하는데 소모되는 비용으로 정의되며, 검출된 취약점별 소모되는 비용의 합으로 구해지게 된다. 아래는 취약점 처리에 소모되는 총 비용을 구하는 수식이다.

$$TVTC = \sum_{i=1}^{n} VTC_i \quad (2)$$

$TVTC$ = 전체 취약점 처리 소모비용 합
VTC_i = 우선순위가 i인 취약점 처리 소모비용 합
n = 검출된 취약점 항목의 수
i = 취약점 처리 효율 우선순위

Equation (2)에서 합을 구하는 순서는 취약점의 처리 효율지수 우선순위를 고려하여, 우선순위가 높은 항목부터 계산하여 구하게 된다.

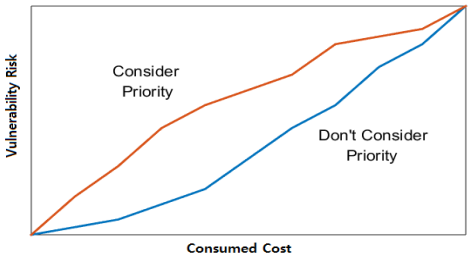

Fig. 2. Consumed Cost per Risk Compare Graph

위의 Fig. 2는 처리 효율 지수 우선순위를 고려하였을 때와, 고려하지 않았을 때의, 처리 효율을 비교한 그래프이다. 그림에서 X축은 취약점을 처리하는 비용의 합을 나타내고, Y축은 처리된 취약점의 위험도의 합을 나타낸다. 위 그림에서 보면 우선순위를 고려하였을 때, 같은 비용을 소모하더라도, 더 많은 위험도가진 취약점을 처리하는 것을 확인할 수 있다.

4.4 취약점 처리 효율 비교

취약점의 처리는 투입 비용에 따라서, 발생한 취약점을 전부 처리할 수 있는 경우와, 전부 처리할 수 없는 경우 2가지로 나뉜다. 앞 장에서 계산한 처리 효율 지수 우선순위를 고려하여 구한 전체 취약점의 소모비용은 취약점을 전부 처리할 수 있는 상황에서는, 효율성을 얻기 힘들지만, 취약점을 전부 처리할 수 없는 경우에는, 취약점 처리에 효율성을 얻을 수 있다.

1) 취약점을 전부 처리할 수 있는 경우

취약점을 전부 처리할 수 있는 경우는, 전체 취약점 처리 소모비용 보다 투입 비용이 크거나 같은 경우이다. 아래 Fig. 3은 취약점 처리 소모비용과, 취약점 처리 투입비용을 비교한 그림이다. Fig. 3에서 Consumed Cost는 발생한 취약점의 처리에 소모되는 비용의 합을 나타낸 것이다. 즉 Consumed Cost의 가장 오른쪽 점은 취약점 전체를 처리했을 때의 소모되는 전체 비용을 의미한다.

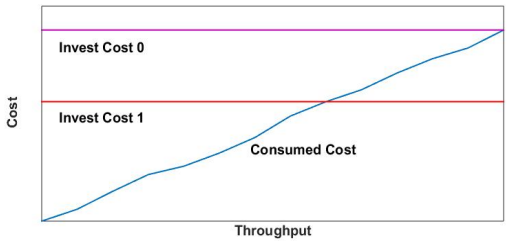

Fig. 3. Case : Consumed Cost and Invest Cost Compare Graph

Fig. 3에서 Invest Cost 0을 보면, 취약점을 처리하는데 소모되는 비용의 합이 취약점을 처리하는데 투입되는 비용과 같은 것을 볼 수 있다.

이 경우의 처리 효율 지수를 우선순위를 적용하였을 때와, 적용하지 않았을 때의 처리 효율을 비교하면, 아래와 같은 Fig. 4를 얻을 수 있다.

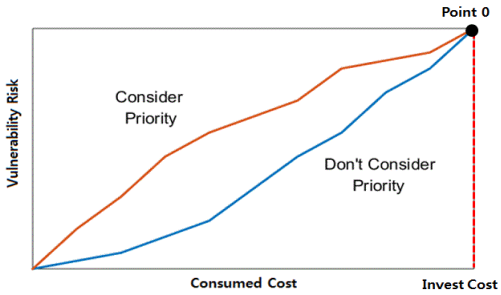

Fig. 4. Treatment Efficiency Compare When All of Vulnerability Treat

Fig. 4에서는 전체 취약점 처리 소모비용과 투입 비용이 같은 경우이기 때문에, 그래프의 X축의 가장 우측 값에 투입 비용이 그래프가 표현하고, 우선순위 고려 그래프와 우선순위 미 고려 그래프와의 접점을 Point 0으로 표현하였다. Point 0의 값을 보게 되면, 우선순위 고려 그래프의 값과 우선순위 미 고려 그래프에서 같은 값을 가지는 것을 알 수 있으며, 즉 처리 효율이 우선순위를 고려하였을 때와 고려하지 않았을 때가 같은 것을 알 수 있다.

2) 취약점을 전부 처리할 수 없는 경우

취약점을 전부 처리 할 수 없는 경우는, 전체 취약점 처리 소모비용 보다 투입비용이 작은 경우이다. Fig. 2에서 Invest Cost 1을 보면, 전체 취약점 처리 소모비용의 합 보다 투입 비용 이 작은 것을 확인할 수 있다. 이 경우에는 발생한 모든 취약점 처리가 불가능하기 때문에, 주어진 비용 내에서 최대의 처리 효율을 낼 수 있는 방법을 사용해야 한다. 본 논문에서는 처리 효율을 비용 당 위험도로 정의하였기 때문에, 같은 비용을 투입하더라도, 최대의 위험도를 처리할 수 있어야 한다. 아래 Fig. 5는 처리 효율 우선순위를 고려하였을 때와, 고려하지 않았을 때의 취약점 처리 소모비용 당 위험도를 비교한 그래프이다.

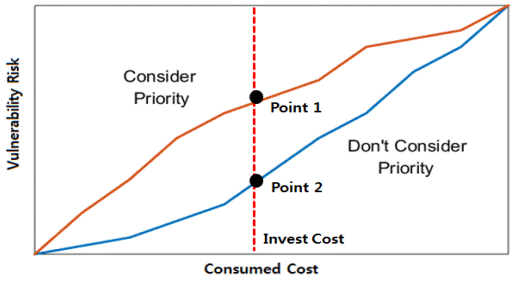

Fig. 5. Treatment Efficiency Compare When Part of Vulnerability Treat

Fig. 5는 취약점 처리 투입 비용 보다 취약점 처리 소모비용이 작기 때문에, 그래프의 X축의 가장 우측 값을 제외한 X 값에 투입 비용 그래프를 표현할 수 있다. Fig. 4 접점 Point 1과 Point 2를 보면, Point 1은 처리 효율 우선순위를 고려한 그래프와의 접점이고, Point 2는 처리 효율 우선순위를 고려하지 않은 그래프와의 접점이다. 이 두 개의 접점을 비교하면, 취약점 처리에 같은 비용을 투자 한다고 가정 하였을 때, Point 1이 가지는 Y의 값이 Point 2가 가지는 Y 값 보다 큰 것을 확인할 수 있다. Fig. 4에서 Y 값은 취약점이 가지는 위험도를 의미하기 때문에, 같은 비용을 투입하더라도, 처리 효율 우선순위를 고려하였을 때, 더 큰 위험도를 처리 하는 것을 확인할 수 있다. 처리 효율 우선순위를 고려할 시 더 큰 위험도를 처리하게 되므로 취약점 처리에 보다 효율적이게 된다.

5. 결 론

본 논문에서는 투입 비용 내에서 최대의 위험도를 처리하는 것을 목표로 하여, 구현 단계에서 정적 분석 툴을 이용하여 취약점을 검출하고, 검출된 취약점에 대한 처리 제어 방법을 제시하였다. 취약점 처리 제어에는 비용 당 위험도를 통하여 정의된 처리 효율지수를 사용하였으며, 투입 비용에 따른 최대 위험도를 처리할 수 있었다. 하지만, 처리 효율을 계산하는 과정에서 사용된 취약점의 위험도와 처리 비용은 CWE TOP25의 정보만을 사용하여 정의하였기 때문에, 개발하는 소프트웨어의 특징을 위험도와 비용에 반영하기에는 한계가 존재한다. 이러한 문제를 해결 하기 위해서는 소프트웨어의 특징을 반영하는 할 수 있는 위험도와 비용의 측정 방법이 필요하게 되었지만, 소프트웨어의 모든 특성을 고려하여 취약점의 위험도와 처리 비용을 산출하기에는 어려움이 따르고 있다. 이러한 문제를 해결하기 위해서 소프트웨어 특성에 따라서, 변경되는 취약점의 위험도와 비용을 측정할 수 있는 방법에 대한 연구가 진행되어야 한다. 이러한 방법이 확보될 경우에는, 소프트웨어의 목적과 특성에 맞는 취약점의 위험도와 처리 비용이 정의가 가능하게 된다. 취약점 처리 제어에 소프트웨어의 특징을 반영한 취약점의 위험도와 처리 비용을 사용할 수 있다면, 보다 정확한 처리 효율을 얻을 수 있을 것으로 기대된다.

References

[1] Gartner, Now is the time for security at Application Level [Internet], https://www.sela.co.il/_Uploads/dbsAttachedFiles/GartnerNowIsTheTimeForSecurity.pdf.

[2] Department of Homeland Security, Practical Measurement Framework for Software Assurance and Information Security [Internet], http://buildsecurityin.us-cert.gov/.

[3] NIST, The Economic Impacts of Inadequate Infrastructure for Software Testing, 2002.

[4] M. G. Choi and M. J. Jeon, “Analysis of Methodologies for Security Development Lifecycle for Security Enhancement System,” *KIMS Spring Symposium*, 2010, pp.418-425. 2010.

[5] Microsoft, Introduction to the Microsoft Security Development Life cycle [Internet], http://www.microsoft.com/security/sdl.

[6] NIPA Software Engineering Center, Software Engineering Withe Book, ch.3, pp.176-183, 2013.

[7] Jovanovic, Nenad, Christopher Kruegel, and Engin Kirda, “Pixy: A static analysis tool for detecting web application vulnerabilities,” in *Security and Privacy, 2006 IEEE Symposium on*, pp.258-263. IEEE, 2006.

[8] Sung min Ahn, Min Sik Jin, and Kyu Jin Cho, “Detecting Software security vulnerability with of Software Security Vulnerabilities,” *Communication of the Korean Institute of Information Scientists and Engineer*, Vol.28, No.2, pp.32-36, 2010.

[9] Mitre, CWE./SANS Top 25 [Internet], http://cwe.mitre.org/top25/.

[10] Mitre, CWSS [Internet], http://cwe.mitre.org/cwss/cwss_v1.0.1.html.

[11] Leung, Hareton and Zhang Fan, “Software cost estimation,” *Handbook of Software Engineering*, Hong Kong Polytechnic University, 2002.

[12] S. K. Choi and E. H. Choi, “Study on validating proper System Requirements by using Cost Estimations Methodology,” *KCSA Transactions on Convergence Security*, Vol.13, No.5, pp.97-105, 2013.

[13] HP fortify [Internet], http://www8.hp.com/h20195/v2/GetPDF.aspx/4AA5-7039ENW.pdf.
[14] Sung hae Kim, Jin ho Joo, Gunsoo Lee, and Gi hwon Kown, "Implementation of Code Vulnerabilities Checker for Secure Software," in *Proceedings of the Korean Society For Internet Information*, Vol.2010, No.6, pp,605-608, 2010.

이 기 현
e-mail : qkqn147@gmail.com
2015년 단국대학교 컴퓨터과학과(학사)
2015년~현 재 단국대학교 컴퓨터학과 소프트웨어보안 석사과정
관심분야 : 소프트웨어 공학, 소프트웨어 보안

김 석 모
e-mail : seokm0@naver.com
2014년 홍익대학교 컴퓨터정보통신공학과(학사)
2014년~현 재 단국대학교 컴퓨터학과 소프트웨어보안 석사과정
관심분야 : 소프트웨어 취약점 탐지, 테스팅, 네트워크 보안

박 용 범
e-mail : ybpark@dankook.ac.kr
1985년 서강대학교 전자계산학과(학사)
1987년 폴리테크닉대학교 전자계산학(석사)
1991년 폴리테크닉대학교 전자계산학(박사)
1993년~현 재 단국대학교 전자계산학과 교수
관심분야 : 지능형 소프트웨어 공학, 소프트웨어 보안, 소프트웨어 품질

박 제 호
e-mail : dk_jhpark@dankook.ac.kr
1985년 서강대학교 전자계산학과(학사)
1993년 폴리테크닉대학교 전자계산학(석사)
2001년 폴리테크닉대학교 전자계산학(박사)
2003년~현 재 단국대학교 전자계산학과 부교수
관심분야 : 이미지 프로세싱, 데이터베이스, 검색 알고리즘