Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 13, no. 7, November 2017, pp. 1101-1110
DOI: 10.23940/ijpe.17.07.p13.11011110

Software Trustworthiness Static Measurement Model and the Tool

Yan Li, Zhiqiang Wu, Yixiang Chen*

Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, 20062, China

Abstract

Software trustworthiness has become one of the prominent studies in software quality assurance, in which the trustworthiness measurement is the primary topic. Compared with the method to evaluate the software development process, we measure to what extent the entity of software better fits users' requirement. In this paper, we propose a bottom-up method of software trustworthiness measurement based on the source code. First, for the trustworthiness measurement of attributes, a comprehensive model is proposed. Second, the validity and stability of the model are verified by Monte Carlo simulation. Finally, the proposed method is developed based on the open source static detection tool for Cppcheck, which forms the software trustworthiness static measurement tool for TSMT.

Keywords: software trustworthiness; trustworthiness measurement; cppcheck

(Submitted on July 25, 2017; Revised on August 30, 2017; Accepted on September 15, 2017)

(This paper was presented at the Third International Symposium on System and Software Reliability.)

1. Introduction

Due to the gradual expansion of software, software defects are growing and software quality is difficult to predict and control. The availability and trustworthiness of key domain application are especially difficult to guarantee for software. These are the important problems in the field of embedded system and put forward higher requirements for the software trustworthiness [4]. How to ensure software trustworthiness is the core and difficult scientific problem in software engineering field [11]. To solve this problem, we first need to solve the relevant work about how to measure the software trustworthiness.

Aiming at the problems of high complexity, large scale and uncertain demand for the field of embedded system, Mei [7] puts forward a method to compute the trustworthiness of the attribute based on the aging problem of trusted evidence, and constructs the trusted evaluation system of the CPS (Cyber-Physical Systems) in combination with the weight distribution to construct method for software trustworthiness. Y Huang [2] puts forward a trustworthiness method based on evidence reasoning for embedded software. Since trustworthiness metrics is supported by related evidence, the theory of trustworthiness evaluation based on evidence is universally recognized in the process of software development.

C-language programs commonly used in the field of embedded system, this paper puts forward a static measurement model of software trustworthiness based on untrusted evidence. We put forward the definition of the untrusted evidence. According to the different data types for the untrusted evidence, we come up with a different model of the trustworthiness rank metric, which corresponds to the trusted attribute, and the number of the untrusted evidence of the attribute in the CWE (Common Weakness Enumeration) flaws storehouse. The attribute is classified by a trusted level, and then a metric model is presented based on the untrusted evidence. The validity and stability of the model are verified by Monte Carlo simulation.

In this paper, the software metrics tool TSMT based on Cppcheck is designed and implemented, which can call Cppcheck to inspect the program. The extended interface based on Cppcheck provides the custom rules on the basis of the

* Corresponding author.
E-mail address: yxchen@sei.ecnu.edu.cn

original inspection rules, so that the tool can detect more untrusted evidence. The tool implements the model proposed in this paper, and perfects the software trustworthiness metric system based on the untrusted evidence.

The rest of the paper is organized as follows. Section 2 mainly introduces the untrusted evidence and its trusted analysis method, which lays the foundation for the attribute trustworthiness measurement. Section 3 introduces a kind of metric model for trustworthiness attribute based on trusted and untrusted evidence, and weights the existing metric model for software trustworthiness based on attribute, and verifies the validity of the model by simulation. Section 4 introduces metric tool for the software trustworthiness called TSMT based on Cppcheck. The last section presents the conclusions and looks forward to the next research work.

2. Untrusted Evidence

2.1. Definition of untrusted Evidence

In this paper, we use the definition of software trustworthiness in literature [3]. According to the definition, we get the definition of untrustworthiness from the reverse thinking. Software untrustworthiness refers to deviating from the user's expectations for the dynamic behavior and result of the software system, or cannot provide continuous service when suffering from interference [5].

Untrusted evidence refers to a program element or program unit hidden in the source code that leads to software untrustworthiness, as shown in the six-tuple:

$$Evidence = < Description, Type, Property, T\text{-}Value, T\text{-}Level, Attributes >$$

In the above six-tuple, Description is the description of the evidence; Type is the data of the evidence; Property indicates the trustworthiness of the evidence to be satisfied; T-value indicates the measure value of the untrusted evidence; T-level represents the level of trustworthiness for evidence; Attributes indicates a trusted attribute that represents the effect of evidence.

The program elements and the program units in the definition refer to the relevant definitions in [9]. It is believed that program elements are variables, constants, data types, algorithmic expressions, logical expressions, empty statements, assignment statements, sequential statements, conditional statements, circular statements, and declarations of variables. The effect of untrusted evidence on software trustworthiness can be decomposed into one or more trusted attributes. According to the list of the program elements, program units and attributes corresponding to the work of Tao et al. [10]. We construct the corresponding relational table of attributes related to each piece of untrusted evidence in Table 1.

Table 1. Untrusted evidence with attributes for corresponding list

Untrusted evidence	Trustworthiness attribute
Improper UI	Functionality, Maintainability
Improper break	Functionality, Reliability
Data overflow	Functionality, Survivability
Division by zero	Functionality, Reliability
Dimension unity	Functionality, Maintainability
Logical error	Functionality, Maintainability
Improper process scheduling	Functionality, Reliability
High complexity	Reliability, Maintainability
Lack annotation	Maintainability

A set of untrusted evidence may affect a number of software properties at the same time, and we only give the trustworthy evidence to affect the main attributes. Users can expand on the basis of actual needs.

2.2. Classification

Our goal is based on the impact of untrusted evidence for software. According to the extent to which the software is affected by untrusted evidence, we divide the trustworthy level of the untrusted evidence from high to low. When the trustworthiness level become much higher, it is more difficulty to improve. Therefore, the higher the trustworthiness level, the much stricter the quantity and influence degree of the untrusted evidence is required. So we propose a classification model of untrusted evidence. In the grading model of untrusted evidence, classification range is not equal. It approximates the ratio of gold division to decrease when we increase the value interval of a trusted level from the lowest. The division contents are shown in Table 2. The method of calculating the ratio of gold division is as follows.

$$0.45 \cdot (\sqrt{5}-1)/2 \approx 0.25,$$
$$0.25 \cdot (\sqrt{5}-1)/2 \approx 0.15,$$
$$0.15 \cdot (\sqrt{5}-1)/2 \approx 0.10,$$
$$0.10 \cdot (\sqrt{5}-1)/2 \approx 0.05.$$

Table 2. The classification of trustworthiness of untrusted evidence

Trustworthiness Level	Definition	Trustworthiness Degree
V	Totally trustworthy	1.0
IV	Trustworthy	0.95
III	Partially trustworthy	0.85
II	None trustworthy	0.70
I	Software cannot run or run incorrectly	0.45

2.3. Trusted Analysis Model

There are many types of untrusted evidence in the procedure, and a single trustworthiness analysis method can hardly cover all the untrusted evidence. Therefore, the data types of the untrusted evidence are divided into two kinds: the Boolean type (represented by "B"), the numeric type (expressed in "D").

The measure distribution of Boolean untrusted evidence is a discrete distribution, and it's only 2 measures to "1" or "0". Generally, the "0" indicates that the untrusted evidence does not satisfy the trustworthiness, and the "1" indicates that the untrusted evidence satisfies the trustworthy nature. When the untrusted evidence satisfies the trustworthy nature, the procedure statement represented by the evidence is trustworthiness. So the measure value of the untrusted evidence is 1, and the level of the untrusted evidence is the maximum level which can be obtained by the type of the change. The procedural statement represented by the evidence is not trustworthiness when the untrusted evidence doesn't satisfy the trustworthy nature. So the measure value of the untrusted evidence is 0, and the untrusted evidence is Grade I. The maximum level here is not necessarily the maximum level in a hierarchy, but the maximum degree of trustworthiness that the evidence of untrustworthiness can achieve. An example of untrusted evidence show " if-else statements do not match " the form of Table 3.

Table 3. Untrusted evidence " if-else statements do not match " sample

Description	Type	Property	T-Value	T-Level	Attribute
If-else statement mismatch	B	Whether the conditional Combination statement matches	0	3	functional

The model is used to denote the level of Boolean untrusted evidence.

$$T-Value = \begin{cases} 1, if\ Evidence.Performance \vdash Evidence.Property \\ 0, if\ Evidence.Performance \nvdash Evidence.Property \end{cases}$$

$$T-Level = \begin{cases} highest\_level, & if\ T-value = 1 \\ \text{I}, & if\ T-value = 0 \end{cases}$$

The value of numerical model of untrusted evidence is number, which can be continuous or discrete. We divide into different grades in the metrical value range corresponding different interval. For example, what's the complexity about loop in the untrusted evidence. According to MACCABA, [6] propose the relationship between the cyclomatic complexity and error; we divide the value range of the cyclic complexity in Table 4. We give an example for untrusted evidence in Table 5 if the complexity of the loop is 20. In this example, although the metric is divided into four intervals, if the trustworthiness of the complexity is level IV and also satisfies the V-level definition, the trustworthiness of untrusted evidence is still level IV.

Table 4. Relationship between cyclic complexity and error rate

Complexity of loop	Error rate	Trusted level
1-10	5%	IV
20-30	20%	III
>50	40%	II
>100	60%	I

Table 5. Untrusted evidence " complexity of loop " sample

Description	Type	Property	T-Value	T-Level	Attribute
Complexity of Loop is 20	D	Whether complexity of Loop is high	20	3	Reliability Maintainability

This is the model to represent the grade judgment of the numerical type of untrusted evidence.

$$T-Level=\begin{cases} T\text{-}Value\ corresponding\ interval\ level, & if\ T-value\ falls\ within\ interval \\ T-Value\ lower\ adjacent\ interval\ level, & if\ T-value\ not\ in\ any\ interval \\ highest\_level,\ if\ T\text{-}value\ better\ than\ all\ interval\ optimal\ values \end{cases}$$

Note: The highest_value here is not the highest level of untrustworthiness, but the highest level of evidence of untrustworthiness.

2.4. The statistic of the evidence of CWE

CWE (Common weakness enumeration) is a software community project designed to create a software flaw enumeration class to better understand software bugs and create automated tools to identify, fix, and prevent such defects. For common software flaws, the CWE organized a list of more than 1500 different vulnerability samples in the real world, and form an article for PLOVER [1] to public use. From this article, we sorted out 25 kinds of software flaws and included 258 flaws entry. The defect refers to the software code in the error or vulnerable to vulnerable weaknesses, and belongs to the category of untrusted evidence. According to the detailed description of PLOVER, each type of software flaws is mapped to software attributes, and the corresponding untrusted evidence is given. According to the data of untrusted evidence, we respectively give model of untrusted evidence classification. Finally, we apply the model of untrusted evidence to flaws library of the CWE. We give a relational table of trustworthiness attribute for each CWE flaw in Table 6.

Table 6. Relational table of trusted attribute for each CWE flaw

Sort	Description	Trustworthiness attribute
BUFF	Buffer overflow	Functionality, Reliability, Survivability
SVM	Structure problems	Functionality, Maintainability, Survivability
SPEC	Special element issues	Functionality, Maintainability
SPECM	Operational problems of special elements	Functionality, Reliability, Maintainability
PATH	Traversal problems of path	Functionality, Maintainability
CCC	Normative problems of code	Functionality, Reliability, Maintainability
INFO	Information management	Functionality, Maintainability, Survivability
RACE	Resource competition	Functionality, Reliability, Survivability
PPA	Authority problems	Survivability
HAND	Error handler	Functionality, Reliability, Survivability
UI	Design problems	Reliability, Survivability
INT	Error interactions	Functionality, Reliability, Survivability
INIT	Initialization errors	Functionality, Reliability
RES	Resource management problems	Functionality, Reliability, Survivability
NUM	Numeric problems	Functionality, Reliability, Survivability
AUTHENT	Authorization problems	Reliability, Maintainability, Survivability
CRYPTO	Encryption errors	Reliability, Survivability
RAND	Random and predictive problems	Reliability, Survivability
ERS	Exception handing problems	Functionality, Reliability
VER	Data verification problems	Reliability, Survivability
ATTMIT	Attack defense problems	Reliability, Survivability
MAID	Change errors of immutable data	Functionality, Reliability, Survivability
MAL	Insertion of vicious codes	Functionality, Reliability, Survivability
CONT	Sensitive data problems	Reliability, Survivability
MISC	Other problems	Functionality, Reliability, Survivability Maintainability

Figure 1 gives the number of untrusted evidence for each software flaw category, the horizontal axis in English abbreviation for each category of software flaws, the number of weaknesses contained in the vertical shaft for each flaw category, or the number of untrusted evidence.

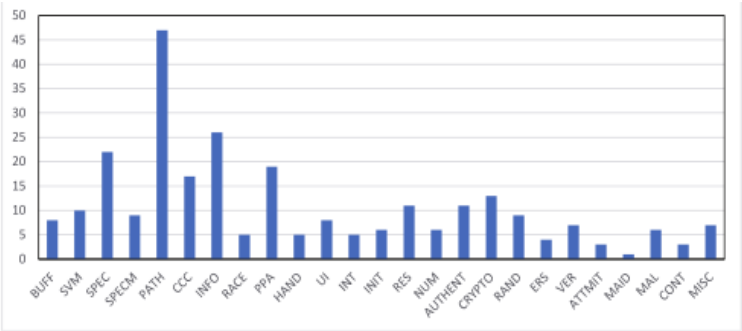

Figure 1. The distribute of the untrusted for the corresponding flaws

3. Comprehensive Trustworthiness Metric Model

3.1. Model

In the previous National Fund project, we make a deep research on the software trustworthiness metric model. A trustworthy metric model based on program slicing complexity can be widely applied to software entity with clear code. However, this model is only for the trustworthiness of the evaluation of the program, not considering the impact of untrusted evidence on the program. Therefore, we propose a comprehensive and trustworthy metric model about attribute based on the original model. The main idea of the model is that trustworthiness of the software entity is determined by the trustworthy nature of the implementation and the evidence of the untrustworthy in the procedure.

First of all, the different degree about trustworthiness impacts overall trustworthiness for software trustworthiness. So, trustworthy measure of software trustworthiness is classified as different level in Table 7. According to the actual situation, the minimum attribute level is 0.45, so it is necessary to ensure that the trustworthiness of each attribute is kept at a good level to ensure that the software is trustworthy.

Table 7. Software trustworthy attribute division

Degree	Trustworthiness range
No impact	>0.95
Little impact	0.85-0.95
Moderate impact	0.70-0.85
Serious impact	0.45-0.70
No run	<0.45

Considering the influence of the untrusted evidence on the trustworthiness attribute, our comprehensive model meets the following criteria and trustworthy attribute of T_{y_i} satisfy: $0 < T_{y_i} \le 1$.

- The trustworthiness of attributes increases when the trustworthiness of untrusted evidence increases.

$$\frac{\partial T_{y_i}}{\partial a_i} > 0 \tag{1}$$

T_{y_i} represents the trustworthiness of attribute of y_i, and a_i represents the trustworthiness of every untrusted evidence.

- The attribute trustworthiness decreases when the number of untrusted evidence increases.

$$\frac{\partial T_{y_i}}{\partial n_i} < 0 \tag{2}$$

n_i represents the number of untrusted evidence of in the y_i.

According to the second rule, it represents the greater the number of untrusted evidence and the lower the trustworthiness attribute. By the classification of trustworthiness attribute, trustworthiness of attribute is reduced to a lower

level, which should contain more untrusted evidence by the classification of trustworthiness attribute. The difference between levels increases gradually, and the difference between levels of untrusted evidence increases gradually. Therefore, we require the number of n_i as the independent variable, and the U_{y_i} represents dependent variable. The curve of the U_{y_i} with the n_i is roughly as shown in Figure 2. the horizontal axis represents the number of untrusted evidence; the vertical axis represents the trustworthiness attribute. The relationship function of trustworthiness attribute and the number of untrusted evidence can be obtained as follow.

$$U_{y_i} = e^{-\lambda_i n_i} \tag{3}$$

Among them, the λ_i is the influence factor for the attribute of i^{th}, and the value range is (0; 1]; n_i represents the number of untrusted evidence that affect the attribute of i^{th} in the program.

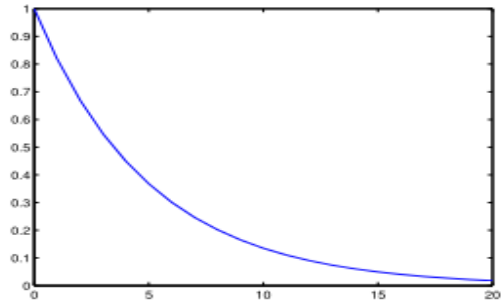

Figure 2. The relationship shows between trustworthiness attribute and the number of untrusted evidence

We take 4 values from the range of λ_i to simulate the relationship between the different trustworthiness attributes. We get the change curve as shown in Figure 3.

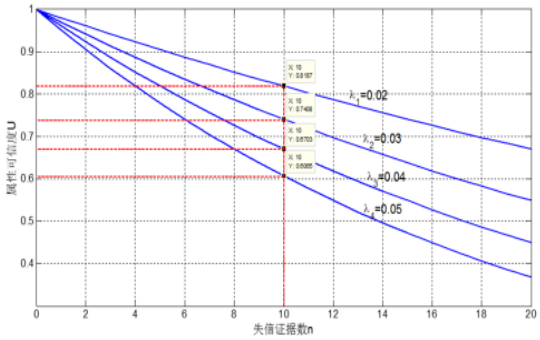

Figure 3. The different λ_i show the relation curve of the corresponding attribute and the number of the untrusted evidence

From the above Figure 3, the greater the influence of the untrusted evidence on the trustworthiness attribute, the faster the trustworthiness of the attribute is reduced with the increasing of the number of the untrusted evidence. It is assumed that each untrusted evidence has the same effect on the program, including the different attributes of the same number of untrusted evidence, and the smaller the weight of the property whose trustworthiness decreases faster. Therefore, the influence factor for λ of the untrusted evidence to trustworthiness attribute is inversely related to the attribute weight for α, and the relation is expressed as follow.

$$\lambda_i = \frac{k_i}{\alpha_i} \tag{4}$$

Among them, α_i represents the weight of i^{th}, and the k_i indicates that the influence factor parameter of the untrusted evidence to the attribute about i^{th}.

The above analysis obtains the relationship between λ and α from the trustworthy relationship for curves of all attributes, and then is determined the range of k_i by studying the trustworthiness of the attribute. Firstly, we convert the trustworthy attributes for calculation model as follow:

$$U_{y_i} = e^{-\frac{1}{\alpha_i} \cdot k_i \cdot n_i} \tag{5}$$

Assuming weights of attribute about the α are known, and when the k_i takes different values, the trustworthy degree of the same property changes as shown in Figure 4.

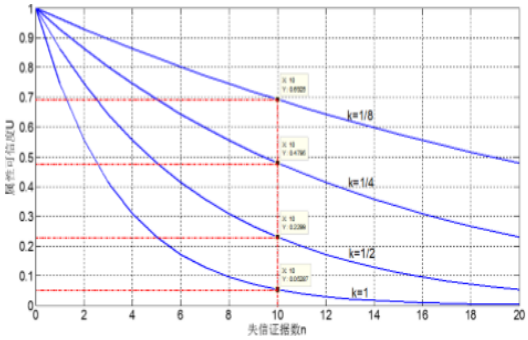

Figure 4. The different k_i show the curve of trustworthy degree of the attribute

From the above Figure 4, with the decrease of value of the k, the trend of trustworthy degree of attribute becomes slow down and gets closer to the linear relation. For the same untrusted evidence, with a greater value of the k, the trustworthy degree becomes lower. If we want to achieve the same trustworthy level, when value of the k increases, the untrusted evidence will be reduced.

3.2. Simulation and Analysis

For the model proposed in this paper, a large number of examples need to be validated, in which the acquisition of untrusted evidence is a difficult problem in the process of verification in software, so we use a large number of data simulations to simulate the number of untrusted evidence of each attribute, the results can be used to observe the stability of the model. Monte Carlo simulation is a method to study its distribution characteristics by setting up a stochastic process, generating time series iteratively, and calculating parameter estimates and statistics [8]. Mathematic is a scientific computing software that combines numerical and symbolic computing engines, graphics systems, programming languages, and text systems that are good for advanced connections with other applications.

The simulation parameters and results are described in detail below.

The number of y_i is 4, and the evaluation value of each attribute is 1, namely: $y_i \in (0, 1]$;

The weight of the attribute is α_i: $\alpha_1 = 0.294$, $\alpha_2 = 0.382$, $\alpha_3 = 0.169$, $\alpha_4 = 0.155$;

The weights of the trustworthy elements and the untrusted elements in the attribute are set to 0.5;

We use 0.001 as the basic unit, and randomly generate 100,000 sets of data to simulate the model in the [0, 1] interval of the cumulative number of occurrences (in the graph by point), and the simulation statement is as follows:

The simulation results show the Figure 5. The horizontal axis represents the software's trustworthiness, and the number of times the vertical shaft represents the trustworthiness attribute.

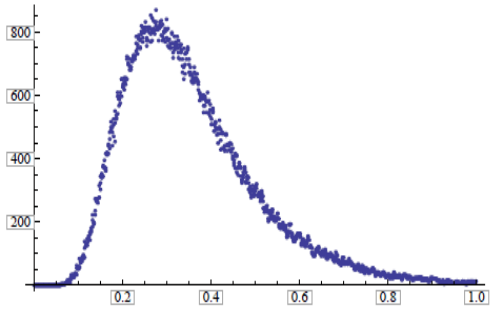

Figure 5. Model simulations

The results of the simulation under the condition of different untrusted evidence show that with the decrease of the untrusted evidence in the attribute, the software trustworthiness is increasing and the number of the untrusted evidence in each attribute need to be strictly controlled. In addition, the shape of the simulation diagram is similar to the normal distribution, which shows the stability of the model.

4. Software Trustworthiness Tool

Based on the models and methods mentioned above, we developed a static predictive software reliability tools based on Cppcheck—TSMT. The tool realizes the evaluation of untrusted evidence, trustworthiness attribute and software trustworthiness in code, and provides the improvement method while detecting the untrusted evidence. This tool embodies the availability of source code oriented software trustworthy metrics.

Cppcheck is developed to use the C++ language, providing a variety of types of error to check for C++, and its checkpoints involve pointers, arrays, memory, and problems in functions. Selecting Cppcheck as the basis for this tool is based on the fact that Cppcheck is an open-source tool, opens the user extension interface, and makes it easy for developers to embed custom rules into Cppcheck. Finally, we can customize the personalized inspection tools.

We introduce the TSMT tool that expands the inspection rules in Cppcheck to enable it to detect more untrusted evidence. So, it supplements measurement system of software trustworthiness based on untrusted evidence. It is shown in Figure 6 as follow.

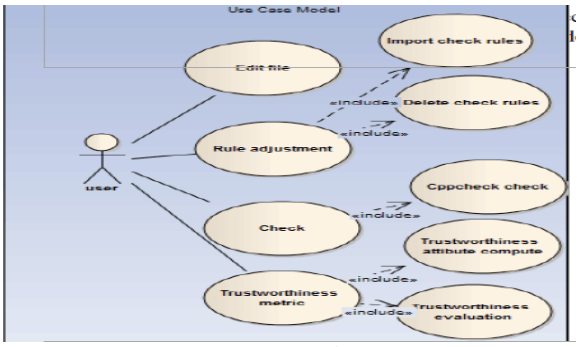

Figure 6. Usecase of TSMT

The tool body can be divided into three parts: user, metric tool, and Cppcheck. In addition to providing a user-friendly GUI in Figure 7, TSMT provides executable file for Cppcheck.exe that can be executed at the command-line prompt, and users can get the results of the check directly using the command line. The trusted metrics tool invokes cmd.exe background execution Cppcheck to get execution results. The specific implementation enters the Cppcheck command in the specified text box through the button call cmd.exe to execute the command in the text box, reads the result from the command line through the StreamReader class, and directly displays it in another text box. Cppcheck can be diversified by adjusting the parameters, introducing the feature into the trusted Metrics tool, and getting a variety of results by entering related commands in the specified text box.

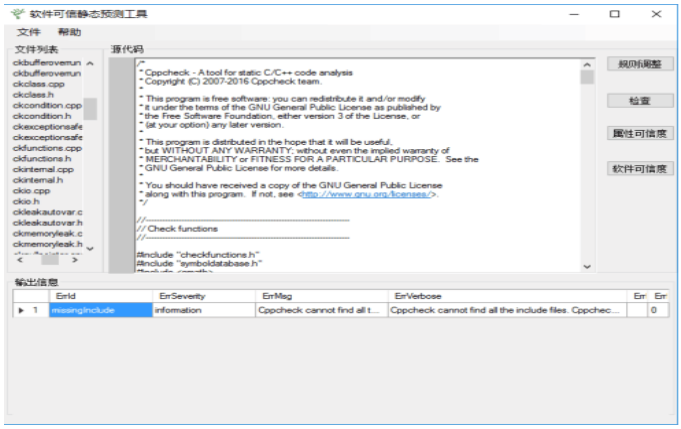

Figure 7. Main View

The tool implements internal call Cppcheck to check the program. The extended interface provided by Cppcheck add custom rule to check on the basis of the original inspection rules, and allow the tool to detect more untrusted evidence.

Webbench is a pressure test tool for website and uses C language development, it includes 600 lines of C. We make trustworthy measurement for webbench.c about the main file for this tool. When adjusting the rules, we click on right-hand function for check and open the interface shown in Figure 8. It enters the Cppcheck execution statement in the text box and displays the results in the text box below. If you do not understand the Cppcheck command and you can click the button for help. The main view shows untrusted evidence in the displayed error lists in Figure 9.

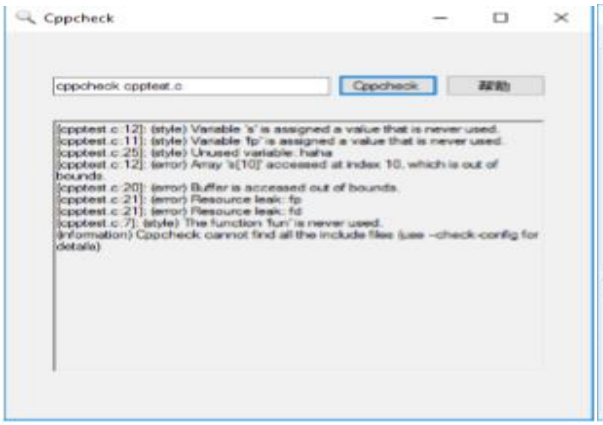

Figure 8. check of Cppcheck

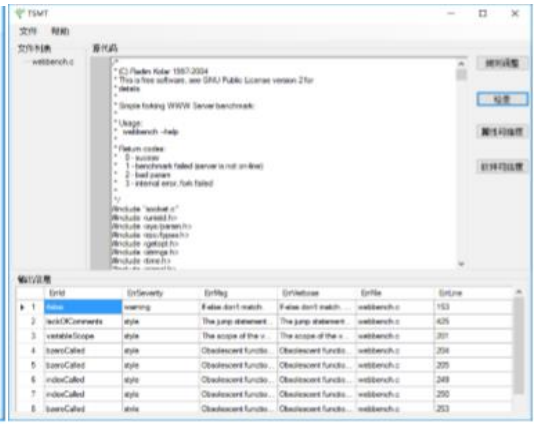

Figure 9. Lists of untrusted evidence

It analyses ten untrusted evidence by detecting and showing that these untrusted evidence belong to procedural normative problem. These problems don't impact the program short term for running software. But there are secure risks, such as "if-else mismatch" problem, which may be attacked and affect the maintainability of the program. The untrusted evidence is detected in source code, although it will affect trustworthiness of the program. It belongs to little impact and the program is relatively trustworthiness.

5. Conclusions

In this paper, the software trustworthiness evaluation is through the detection of untrusted evidence in software source code. Because the characteristics of software source code are easily accessible and the generality of embedded software is commonly used in C language development, this work can be applied to the research of the trustworthiness of embedded software. In the future, we need to perfect the trustworthy metric model and verification by the formal method to enhance the persuasion. Second, to improve the trustworthiness of the measurement tool and achieve a higher automatic degree of software trustworthiness metric, it is necessary to improve the tool.

References

1. S. Christey, “Preliminary List of Vulnerability Examples for Researchers”, 2006.http://cwe.mitre.org/documents/PLOVER.pdf.
2. Y. Huang, X. He, J. Wang, and Z. Lei, “An Evaluation Method Oriented to the Comprehensive Credibility of Simulation Data Source Based on Evidence Theory,” vol. 5, no. 2, 2016.
3. R. Jiang, “A Trustworthiness Evaluation Method for Software Architectures Based on the Principle of Maximum Entropy (Pome) and the Grey Decision-making Method (Gdmm),” Entropy, vol. 16, no. 9, pp. 4818–4838, 2014.
4. K. Liu, “Overview on Major Research Plan of Trustworthy Software,” Bulletin of National Natural Science Foundation of China, 2008.
5. Y. Li and Y. Chen, “A Measurement Model for Trustworthy Software Based on Trusted Evidences,” in International Symposium on System and Software Reliability, 2017, pp. 20–24.
6. T. J. Mccabe, “A Complexity Measure,” IEEE Transactions on Software Engineering, vol. SE-2, no. 4, pp. 308–320, 2006.
7. M. Rong, “A Model for CPS Software System Trustworthiness Evaluation Based on Attributes Classifying”, 2013.
8. A. F. Seila, “Simulation and the Monte Carlo Method,” Technometrics, vol. 24, no. 2, pp. 167–168, 2012.
9. K. Shibata, K. Rinsaka, and T. Dohi, “Metrics-based Software Reliability Models Using Non-homogeneous Poisson Processes,” in International Symposium on Software Reliability Engineering, 2006, pp. 52–61.
10. H. Tao and Y. Chen, “A New Metric Model for Trustworthiness of Softwares,” Kluwer Academic Publishers, 2012.
11. H. Tao and Y. Chen, “A Metric Model for Trustworthiness of Softwares,” in Ieee/wic/acm International Joint Conference on Web Intelligence and Intelligent Agent Technology, 2009, pp. 69–72.