Enhancing the Secured Software Framework using Vulnerability Patterns and Flow Diagrams

Nor Hafeizah Hassan[1], Nazrulazhar Bahaman[2], Burairah Hussin[3], Shahrin Sahib[4]
Faculty of Information Communication and Technology
University Teknikal Malaysia Melaka
Malaysia

***Abstract*—This article describes the process of simplifying the software security classification. The inputs of this process include a reference model from previous researcher and existing Common Vulnerabilities and Exposure (CVE) database. An interesting aim is to find out how we can make the secured software framework implementable in practice. In order to answer this question, some inquiries were set out regarding reference model and meta-process for classification to be a workable measurement system. The outputs of the process are the results discussion of experimental result and expert's validation. The experimental result use the existing CVE database which serves as an analysis when a) the framework is applied on three mix datasets, and b) when the framework is applied on two focus datasets. The first explains the result when the framework is applied on the CVE data randomly which consist mix of vendors and the latter is applied on the CVE data randomly but on selective vendors. The metric used in this assessment are precision and recall rate. The result shows there is a strong indicator that the framework can produce acceptable output accuracy. Apart from that, several experts' views were discussed to show the correctness and eliminate the ambiguity of classification rules and to prove the whole framework process.**

Keywords*—Software secured framework; security classification; software security; common vulnerabilities and exposures

I. Introduction

In software application, it is observed that there are negative consequences when security is compromised. Security can be compromised when there is lack of understanding of the in hand situation. Various terms used for security and it's family, huge numbers of models and framework to refer to, had created confusions to the software practitioner to classify vulnerability that is accurate, consistence and correct.

It is observed that there is a challenge in forming a vulnerability classification scheme due to type of data used. For example, some vulnerability database like Common Vulnerabilities Exposures or CVE is very much using natural language structure but without proper English grammar as given in its web page of Common Vulnerabilities and Exposures: The Standard for Information Security Vulnerability Names. One way to extract the information is by using semantic analysis [1]. However, in security domain, some terms are used differently. For instance, the meaning of buffer overflow is to overwrite the adjacent memory by overrun buffer and is not simply means that buffer is more than full.

Therefore, it is learned that the terms must be specified with related to predefined rules of information security. Another challenge was to formally translate the domain terms into a schema that can be translated to a workable engine to extract the vulnerability given a historical database as debated in [2]. Therefore, this study is to focus on this scenario.

The current vulnerability classifications suffered from multiple dimensions of classifiers. They are either too specific or too complex [3] and [4]. Or they were only for dedicated cases. This leads to disability in performing a detection or protection from newer attack of vulnerability. The understanding of the taxonomy which are also various, requires a formal classification that can be used for generic cases regardless of applications, mobiles, networks or other devices [5]. This study focuses on the research and development of the design, formalization and translation of the vulnerability classification pattern through a framework using common vulnerabilities and exposures data pattern. It is achieved through the usage of syntax and semantic formal representation that not only accurate to produce a simplified set of vulnerabilities patterns but also consistently can be use within other incident cases. The final aim of this study is to measure the accuracy and correctness of the vulnerability classification procedures of algorithm, which already indicates the focal view and depth in security domain.

II. Related Work

The early work on vulnerability classification focused on the knowledge of fault identification. It tried to solve the difficulty of identifying the origin of faults within a software [6],[7].

A. The Origin of Fault

The initial works claimed there is a specific place a fault exist or known as origin. However, the newer research claimed that the place alone cannot be considered as the origin of fault but shall encompass the time it occurred, as debated in [8],[9]. The later work eventually pointed out that the development phase is frequently used as the time the faults are introduced.

B. Fault, Failures, Attacks and Vulnerbailities

The knowledge of this fault identification leads to the terminology emergent of faults, failures, errors and attacks. Shortly, a differentiation of the definition is given in [8] as following. An error is a human action that produces an incorrect result, a fault is an incorrect step, process or data definition in a computer program and a failure is the inability

of a system to perform its required functions within specified performance requirements. As system becomes complex, [10] defined fault as an imperfection that able to result as failure in software.

However, their study focuses on the taxonomy of security terms and did not critically informing enough how the terms can be used to reduce numbers of vulnerabilities. Eventually, another finding is the study of relationship between the terms which perhaps gave new meanings to reduce vulnerability. Few researchers addressed fault as a concrete manifestation of an error within the software. One error may cause several faults, and various errors may cause identical faults which if encountered can cause system failure as deduced by [11] and [12]. However, [9] uniquely elaborates the relationship between vulnerability and error. They defined that vulnerability is an instance of an error, and error can be in specification or development or software configuration. We also learned that [13] in their work introduced the vulnerability term as a characterization of vulnerable states which is distinguish from any non-vulnerable states as in Fig. 1.

Fig.1 shows the relationship of the terms error, fault, failures and vulnerabilities. Perhaps, because of this consensus, later, in Common Vulnerabilities and Exposures: The Standard for Information Security Vulnerability Names, another definition of vulnerability is given as a state in a computing system that either:

- Allows an attacker to execute commands as another user
- Allows an attacker to access data that is contrary to the specified access restrictions for that data
- Allows an attacker to pose another entity
- Allows an attacker to conduct a denial of service

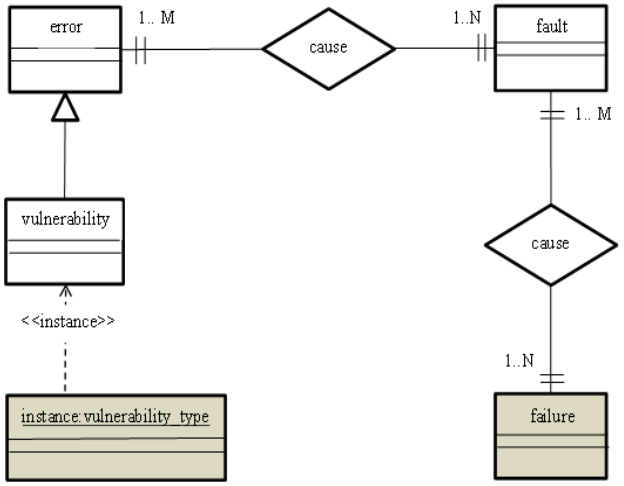

Fig. 1. Relationship of the terms-Error, Fault, Failures and Vulnerabilities

This definition shows, when connecting with definition from Fig. 1, that vulnerabilities are the negative consequences of faults that bring about from one or more errors. And they are always related to external user or attacker who manipulates the vulnerabilities to get access. As a result, a compromised vulnerability can cause a (or various) system failures.

C. Vulnerbailities Classification: The issues

Consequently, the term vulnerability in vulnerability classification reflects an inclusive meaning in related to the reason of any software failures. But, still is the issue of how to classify these vulnerabilities given the fact that vulnerability, so that a developer could know which failure they are related to.

Very often, the common style of classification pro-motes the vulnerabilities to be placed in more than one class, due to the hierarchical style of classification. This technique is called data multiplication. Multiplication of data is the major drawback. For example, if vulnerability x is located into class A and class B, it will return a confusion for later analysis especially for treatment. This will cause an incorrect result. The method were later improved by [14] as discussed in [15, but still not much difference. On top of that, [13] argued the classification of flaws in PA project, RISOS project and in [9] which they meet neither the uniqueness nor the well-defined decision procedure requirement. They suggested that one can view a vulnerability as a containing class and attacks as elements of that class.

In 2005, two researchers from CMU/SEI published their work of new way of classification through technical report in [16]. In order to avoid multiplication issues in mentioned earlier, which using hierarchical style of classification, they suggested the attribute-pair values through object-roles definition of vulnerability. However, the method suffered the level of abstraction and viewpoint. Another researcher, [17] classified the vulnerabilities through characteristic trees, still is a hierarchical style of classification which suffer the tradeoff between different operating system used. In the nutshell, the gap is the vulnerability classification is still an issue where, it could be either too specific to certain system or too complex which, in addition, introduce the ambiguity.

All classification involves the goals and the perspectives. The goals and perspectives demonstrate of how the classification is being seen and carried out. A study on classifications goals and perspectives were done and showed that a trend of classification from as basic as error to as complex as data mining had been carried out. However, the diversity of classification patterns still can be refined as some of them had produced complex patterns with huge number of classes.

D. Vulnerability Classification: The generic Process

The analysis during these study also reveal that each classification scheme by the researchers consist of few generic processes namely as identification, analysis, confirmation and elimination of flaw. The summary of the processes completed by the researchers are as in Fig. 1.

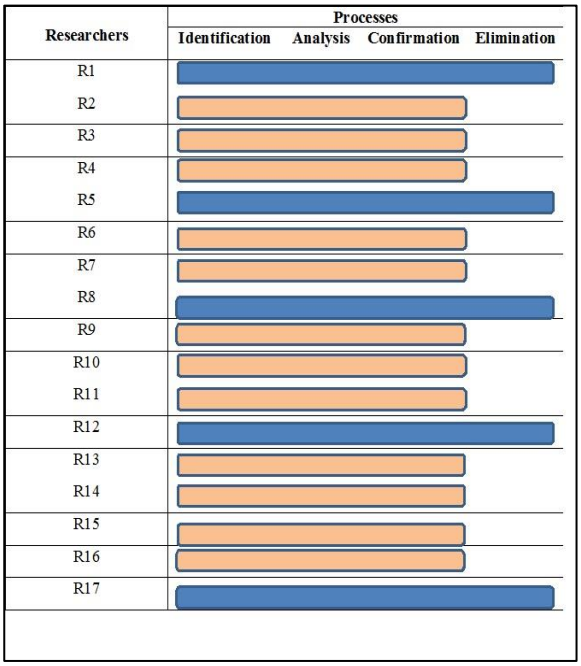

Fig. 2. Vulnerability classification based on process

Fig. 2 shows four generic processes in determining and classifying vulnerabilities: identification, analysis, confirmation and elimination of flaw. The identification process covers from non-unique identification of system resource to specific identification. The analysis process involves the labeling, add-on features and dissemination of identified flaws. The confirmation process is putting the flaw into the appropriate class. And the elimination process informs the reader that the respective perspective also able to assist in reducing the identified flaws. The reasons of having these four generic processes, according to [4], [18] and [19] are to: establish, act upon and maintain relationship about the threat in related with their life cycle.

The development team can gain better awareness of the larger threat viewpoint when this information is shared in advance of an incident, hence reducing the occurrence of next possible bugs and vulnerabilities. And from the analysis, not all past works contained every process. But, here, from the table showed that at least three initial processes must exist in any vulnerability classification scheme - identification, analysis and confirmation. Using these generic processes, many researchers develop their own models or frameworks to reduce the numbers of bugs and vulnerabilities as in [4], [20] and [21]. However, the definition of what kind of framework that should work with vulnerability classification is remained a dispute.

E. Secured Software Framework

A framework is defined as the conceptual structure that dependent to each other to complete specific purposes. The structure could be an artifact of input, output, process, function, or boundaries [22] and [23]. Following the analysis in [13], the definition of framework is given as a sequence of decision procedure which when apply, will classify a state to exactly as one tuple. A decision procedure refers to the application of the function to a specific vulnerable state. Discriminating properties, as embodied in the decision functions, determine classification.

The specific purposes of such a framework are to provide a historical record of the vulnerabilities in a form that software developers can use: a) to anticipate flaws in their systems; b) to describe the vulnerabilities in a form useful for detection; c) to show common characteristics in related flaws for prevention and elimination; and d) to enable a security monitor to detect exploitation (or attempted exploitation) of the flaws.

According to [8], a framework is not simply a neutral structure for categorizing elements. Taking from here, a vulnerability classification is an element in a absolute framework, in which, this framework shall serve as the blueprint working process to do the vulnerability classification despite any aim it is meant for to be delivered. But still, in a context to have secured software.

Hence, the framework to describe the vulnerability classification working process is named as a secured software framework.

A comparison of security framework had been made by [24] which analyzed eleven frameworks from year range 1996 to 2004. The comparison significantly shows the needs for a systematic approach to embed security concerns into software process as early stage. Another work was conducted by [25] to summarize the security dimensions, such as cause, impact, and location, encountered in security frameworks.

However, there is still lack of research done of how to integrate secured process operation in software development process [26]. A comparison work on this secured software framework was done in [27]. The summary is given as in Table 1.

The researchers review had revealed that those frameworks start their security consideration as early as requirement stage as depicted in Table 1.

TABLE I. THE STAGES FOCUSED IN SOFTWARE SECURITY PROCESS FRAMEWORK

Software security testing framework	Stage	
	Requirement	Design
KAOS (Security Extension)	✓	✓
MDS (Model Driven Security)	Not stated	✓
i* framework	✓	Not stated
ST (Secure Tropos)	✓	Not stated
SQUARE	✓	✓
SREP	✓	✓
SRE	✓	✓
TM (Threat Modeling)	✓	✓

The summary demonstrates that in the mentioned security frameworks, security is a concern as early as during the requirement stage as in [28] and [29]. They highlighted the needs for a standardized methodological approach that taking into account security elements from the earliest stages of development till the completion.

F. Measurement

Even though [30] highlight the quality attributes of CIA-triad (confidentiality, integrity and availability) in security studies, this research also emphasized measurement for the algorithm used. The successfulness of the framework is dependent how good the algorithm can extract the required patterns. The extraction-based procedures are measured using:

- precision rate
- recall rate

The precision measures the accuracy of the algorithm and the recall measures the effectiveness of it. However, the correctness of incident case-pattern matching process is measures using expert opinion approach.

III. METHOD

This study focuses on two parts. The first part is to uncover current process of vulnerability classification by identifying the meta-process model and indicate its framework. The second part is to enhance the framework by analyzing the vulnerability classification patterns and confirming ways in accepting affirmation

In identifying meta-process model, task of this phase is to investigate the existing meta-process models of vulnerability by analyzing and synthesizing the core elements with the help of parsing technique in the existing common vulnerability and exposures database to determine whether or not the model satisfies with the objectives.

In analyzing the vulnerability classification patterns, the possible patterns exist in the database by using key aspects in a threat model and the metaprocess model. The list of patterns in threat model are synthesize with the current categorized of threat in the database. A formal classification method through an algorithm is proposed and used to deduce the patterns which will sustain the vulnerability classification framework to accept the subsequent process of affirmation.

And in accepting affirmation is to execute the formal algorithm by applying it within the classification framework and get effectually valid through experimental and expert views.

IV. THE FRAMEWORK

A successful meta-process model highly depends on proper apprehension of their functionalities, contexts and architectures. To achieve this purpose, a reference model technique is used.

A. Identifying Meta-Process Reference Model.

This is an input to the process of generalizing the meta-process model. In this activity, a qualitative study was used to collect, categorize and select the related reference model. The output from this procedure is the formulation of vulnerability classification meta-process. This meta-process is used as the blue-print for the framework. This study is inspired from the work of Linde [31]. In the work, a four steps of classification meta-process was introduced which consist four stages: knowledge of system control structure, the generation of an inventory of suspected flaw, confirmation of the hypothesis and making generalizations of specific flaw instance. The former study was aimed to focus on software attack through generic operating system or OS flaws [6], [7] and [13]. As the study of classification matured, the needs to enhance the methodology arise, for example to support the purpose of doing the classification varies among researches [31]. Therefore, in existing secured software framework, there are at least three stages involved in vulnerability cycle, however four is preferred as the fourth support the mechanism for elimination. The four steps are adapted as the meta-process for vulnerability classification which consists of i) identification, ii) analysis, iii) the confirmation and iv) elimination. On top of that, instead of using class name or perspectives for the countenance, we assign the label of characterization to represent the class showing the significant of a characterization embed in respective class. The label of characterization shall be named after considering the attributes that carry specific value describing the label as discussed in [4]. By answering the attributes with relevant value will help the researcher to present the characterization into a single tier or more. The following subsection elaborates the steps in detail.

B. Analyzing the Process, Activities And Output

The identification process involved identifying the objective of the classification. Then, the researcher refers the previous taxonomy of classification to understand the subsequent philosophy. Within this process, the context (scenario) of each security flaw or attack is analyzed and any available repository is examined to find any match. As it is almost impossible to analyzed the information system flaws as a whole as pointed out by [32], a researcher will sub-task the system by addressing the flaws or attack from the encapsulate platform which either come as an operating system or application based security flaw (or attack). In addition, the nature of the platform (operating system or program) which also referred as target system (or victim), is explored and essential elements are captured.

The output of this process is to confirm any sensitive data that activate an interest to an attack (security objects) and to distinguish the protection mechanism that protects the respective data (control objects) [31].

The analysis process involves labeling and dissemination. These security flaw or attack contexts are further given certain labeling, often to map them with the captured elements of the target system as well as to meet the aim of the research perspective. These labeling contained common shared traits also known as classifiers or attributes of the flaws, which in many cases are very difficult to be precise. To initiate a labeling, a set of questions for the content shall be imposed. [6],[13] and [32]. In the dissemination process, the identified classifiers or attributes that gathered from defined scenario in the activation process are examined, and assigned a formal value. Using this value, the description of the security flaw is

transfer into hierarchical form such as list (flat or non-flat list) and tree (commonly in directed acrylic graph). Very often the hierarchy is going top-bottom or generic (inclusive) to specific [6],[32],[9],[13] and [10]. The selected hierarchical orientation must consider the complexity of security flaw or attack which may act as an amalgamation (the blended form of flaw/attack) and decelerate the process. As the process iteratively matured, the countenances shall update the classifier to suit with the analysis which also referred as categories refinement. The whole process of defining the countenance is also known as specific taxonomy (class syntax) and a good start for a (high-level) class syntax for OS flaws was introduced by [6].

The confirmation of instances process is the placement of any noticeable flaw or attack into the hierarchical form and expected to form a directory of security flaws or attack to assist in next production. A proper directory will indicate the flaw labeling; thus, facilitate the protection and testing process effectively. In common, the dissemination is done in bottom up approach or from a specific flaw towards achieving its generic class.

The elimination process is the detection, elimination and forecasting of faults in the next system development. It covers at least five aspects namely as fault detection, fault prevention, fault tolerance, fault removal and fault forecasting. The process is detailed out in terms of activity and output as simplified as in Table 2.

TABLE II. THE PROCESS, ACTIVITY AND OUTPUT OF A VULNERABILITY CLASSIFICATION FROM CRITICAL REVIEW

Process	**Activity**	**Output**
Identification	Define objective, target system and identify data	Classification objectives, target system specification and a selection of data
Analysis	Select method, construct hierarchy using syntax or semantic and analyze impact	Method selected, hierarchy (list,tree or graph), list of impact
Confirmation	Refinement of the hierarchy	Refined hierarchy
Elimination	Detecting and eliminating the new flaw found	a. Fault detection, b. Fault prevention, c. Fault tolerance, d. Fault removal e. Fault forecasting

C. Conceptualization of Vulnerability Classification Meta-process

Next, we analyze the current approach on vulnerability classification. Each work is determined for their unique processes in conducting the classification; implicitly or explic itly stated in the research work. As a result, it is obviously seen that in every works, the researchers begin with identifying their purposes to align with the results (as identification process). Next, based on the purpose, they analyze the data or any vulnerability sources to determine the appropriate classes (as analysis process). Following to that, the data is assigned into the respective classes, iteratively (as dissemination process) In order to ensure that the assignment is correct, a benchmarking with secondary data such as technical reports or literature review is conducted (as confirmation process). Some classification proposed the solution to the vulnerability while others don't. In elimination process, the mechanism to solve the problem is identified. For example, a fault can be detected and remove. Or a frequent occurrence of fault may lead to fault forecasting and thus precede to fault preventing. The formulation can be presented as in Fig 3.

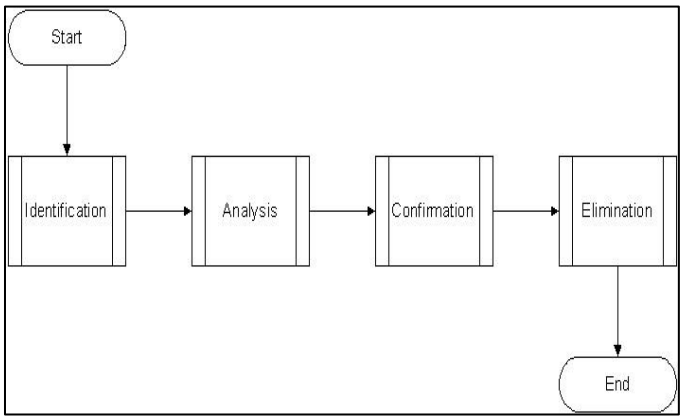

Fig. 3. The Formulation of Vulnerability Classification Meta-process

Fig 3 shows four generic processes involves as a vulnerability classification meta-process - identification, analysis, confirmation and elimination. Next, the details in each of it are explained in the next subsection.

D. The Vulnerability Classification Framework (VulClaF)

This section summarized the study from Table 2: The process, activity and output of a vulnerability classification with Fig 3: The Formulation of Vulnerability Classification Meta-process. We discovered the high-level overall framework for secured software as in Fig 3. In order to detail out each processes, each process can be re-iterate for respective stages-analysis and deployment as in Fig 4.

For example in identification process, involves the activities of defining objective, target user, system and data. And for deployment or output stage, the detail of them is describes as in Fig 4.

However, this framework which was used unconsciously used during classification was suffered from issue of data multiplication that eliminate the uniqueness of a class. Hence, slow down the remedy process.

This issue is mainly due (as in Fig 4) in the Analysis process at activity Produce vulnerability tree. A vulnerability tree has high potential to create redundancy if the classification rules were not formally defined in activity Produce syntax-semantic schema. An enhancement to overcome this is proposed.

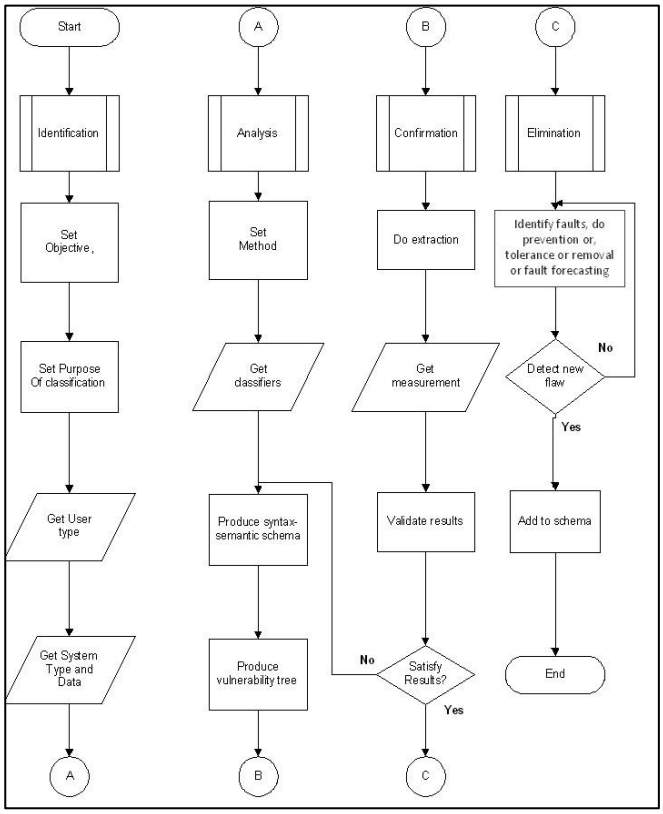

Fig. 4. The Vulnerability Classification Framework (VulClaF)

V. The Enhanced Vulnerbaility Classifcation Framework

The proposed enhanced framework is based on these phases: a) attaining the vulnerability classification patterns and b) accepting affirmation.

A. Attaining Vulnerability Classifcation Pattern

The classification pattern is attained from preprocessed raw data, with the help of the context-free grammar as input. Through this procedure, vulnerability classification patterns are identified after matching process with the domain specific wordlist. However, this procedure involves understanding of existing vulnerability classification trends. Analyzing Existing Vulnerability Classification Trend. This is another input to the process that involved identifying current trend in vulnerability classification.

The output from this procedure is to determine the label of characterization and establish their relationship.

- The Label of Characterization. A numbers of existing vulnerability classification were studied, scoped and analyzed for their similarities elements. It is found out that many terms were used to characterize the attributes or state of threats such as origin, time, OS etc. With that, in this study a generic term to address them is used as: label of characterization and served as partial classifiers because when on their own, they are not sufficient yet to do a good classification. The connection between them must be discovered. The output is to get a set of container with different motives of their existence.
- Discovering Relationship. The sequence of the labels with their cause and effect were later realized as a relationship notation. They were pre-analyzed on a sample of data and tested. The output is to define the cause-effect of each container from label of characterizations. Next, a set of context- free grammar then was established for the purpose to avoid ambiguity in experimental results.
- Context-free Grammar. This is an input to the process of generating the vulnerability classification algorithm. The output consist domain specific schema and domain specific marker.

B. Label of Characterizations

The purposes of classification from past works clearly stated their aims are to do the classification for the purpose of assisting in testing and maintenance as in [4], [33] and [34]. It means that the purposes shall represent the level of abstraction. Level of abstraction implies the attempt how extensive the classification is [34]. The user perspective reflects the aimed users that the classification is made for, like for the usage of software developer or software designer.

The label of characterization depicts the elements or features considered that results as a class in the classification. In [34], the outcomes of classification consist of software development issues, location of flaws in the system and impact of flaws on the system. Each class comprises of second tier of subclasses. In [33], it is classified into seven groups: input validation and representation, API abuse, security features, time and state errors, code quality, encapsulation, and environment.

The research in [4] shows close similarities with the aim of this research which proposes the combination of : i) cause, ii) location, iii) attack vector and iv) impact as the essential label of characterizations for secured process, maintenance and assessment. The cause describes the reason for the existence of the vulnerabilities; the location determines the attacker community that defines the risk level and in turn determines the mitigation strategies. Attack vector defines the attack mechanism used by the attackers and impact describes the degradation of the system performance after an exploit takes place as in [4]. Therefore, this model of [4] is appropriated as the reference model for vulnerability classification within software developer perspective that aim of assisting security process, maintenance and assessment. And based on this model, we scoped the current study of other researchers as in Table 3.

TABLE III. Identifying the Vulnerability Classifiers From Past Works

Purpose	User Perspective	Label of Characterization
To assist security process, maintenance and assessment	Software developer	Cause Location Attack vector Impact

Based on Table 5, we represent the significance of the purposes. Any decision to specify the label of characterization is established based on the purpose of the classification and the target users. The purpose responds to the issue of level of abstraction and the user's perspective counteracts the issues of point of view. However, the connections sequences between the labels remain unclear and the purpose scope is extensive try to cover process, maintenance and assessment – three dimensions of works. Based on the issues addressed, a new label of characterization is proposed to determine the relationship between each class with more specific scope of purpose.

The important aims within secured software are to confirm the presence of security attributes in the software and to increase the delivered reliability of the software to the user as in [35], [36] and [37]. As the vulnerability population is vast, it is impossible to examine each of them individually [37]. However, the secured software focuses on vulnerability exploit that demands the understanding of different states occurred as well as their relationship between each other when compromised. These states describe the behavior of an event that having certain attribute at any given condition. The existing studies describe how they perceive the existence (occurrence) of vulnerability in various ways such as domain, origin, operational and software development life cycle phases. Nevertheless, in software, the occurrence is about writing at least an executable statement of code. Then, regardless of how they are perceived, an occurrence of vulnerability consists of at least four states: i) creation, ii) discovered, iii) exploited and iv) resolved of an executable statement of code. Creation is when the code is created with a specific reason or cause at a definite place or location and still in an idle state. Discovered is when an attacker realizes the particular code as a vulnerability point in the location or application, potential to be compromised and this is a visible state. Exploited is when the attacker uses an attacking mechanism that is appropriate with the cause and location to compromise the vulnerability point and this is the utilize state. Resolved is when a targeted impact of the attack affects the system and an action is required to solve the problem, including appropriate test and this denotes the state of fix. These four states determine the characterization needed during the vulnerability classification model for a secured software framework. In any exploitation of vulnerability, it is essential for an attacker to identify the states of possible exploitable code using an attack mechanism and produce the targeted impact they have expected.

The states are the life cycle events of vulnerability and were represented as a waterfall sequence. However, the relationships between the events are currently not well explained. For example, whether there exists any kind of source and target events, or if there exists bidirectional relationship between them. In order to identify the type of relationship that exists, the study of the vulnerability class must be able to map with the respective states.

These states are further perceived as label of characterization introduced in the Table 4 by accepting the labeling of cause as source root (to refer to the vulnerable code statement), source location as application (to refer to the application container of where the code statement reside), target vector as target (to refer to the effect of attack mechanism use on the vulnerable code), target impact as impact (to refer to the expected result of the exploitation or damage), and this research introduce the notion an additional of attacker (to refer to the user who initiate the attack). As the purpose is to determine the relationship between each label, the component relationship is added as in Table 4.

TABLE IV. THE NEW LABEL OF CHARACTERIZATION AND RELATIONSHIP FOR SECURED SOFTWARE FRAMEWORK

Objective	User Perspective	Labels and relationship of characteristics
To assist secured software framework	Software developer	Label of characterization • Source Root • Source Location • Target Vector • Target Impact
		Relationship between label

Based on Table 4, the pattern for CVE is produced using three steps: labeling, categorizing and producing schema grammar from CVE incidents cases. The first process is labeling the domain specific marker (DSMarker) for each classifier. The markers are used as indicator to pick and group the words into related classifier. The markers are used as input to the second process. The second process is to categorize the words into domain specific wordlist (DSWlist). The purpose of this process is to extract the similarities (domain-based) between the words and form the wordlist. The output of this process is used as an input for the next process, producing the domain specific schema (DSSchema) grammar. The study from the data shows the incidents have obvious marker that indicates which phrase shall belong to a classifier. Fig. 5 shows the DSMarker.

The top level is the primary marker. The second level is the secondary marker, which can be divided into two: the left side of the primary marker and the right side of the primary marker. The third level is the subsequent markers, those that immediately serve the phrase after into the specific classifier denotes as the lowest most level.

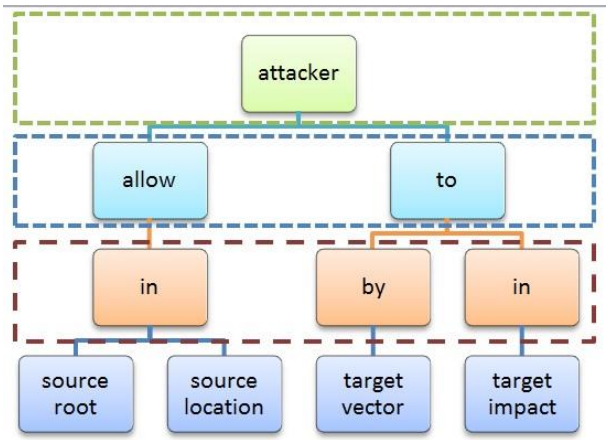

Fig. 5. Showing the generic hierarchical structure of the classifiers

The preliminary study also showed that there are four classifiers: source root, source location, target vector and target impact. Source root is the reason or the antecedent that imply to the impact of vulnerability. The source root can be (but not limited to) buffer overflow, configuration, connection, credential, injection or settings. These conditions are originated from the coding and implementation phases. They are a result of lacking concern of early security imposed in the requirement and design phases either lack of tools used or lack of knowledge about them. Source location is the emplacement or entity where the vulnerability antecedent exists. The place can be (but not limited to) an argument, a string, path, registry, password, host or script. These are places where certain vulnerability may be originated and developed. Interestingly, this analysis found that only few terms share two different classes. The 'mail' and 'directory' could be in class source location and target impact. Therefore, to differentiate them, the keyword 'in' before 'mail' or 'directory' is recognized as subsequent marker to determine them as source location.

Target vector is the magnitude and directions causing damage to occur on target. The magnitude and directions represent any doing that can cause damage such as (but not limited to) change, compromise, check, execute, modify, insert, read or obtain. These actions are showed by the usage of verbs in the pattern. The word magnitude is used as it carries certain weight to cause the damage and the word direction is used to show the aiming point of damage. Target impact is the entity that been affected by the target vector. The effected entity can be (but not limited to) data, database, file, directory, an account, a program or a system. The effect is the outcome or consequences that a target becoming upon a vector is carried out. In order to ease the reference to the classifiers, in the rest of this discussion, each of them are label as: SourceRoot, SourceLocation, TargetVector and TargetImpact. The examples of each classifiers class are as in Table 5.

TABLE V. EXAMPLE OF ITEMS FOR CLASSIFIERS

SourceRoot	SourceLocation	TargetVector	TargetImpact
format	mail	compromise	data
configuration	authentication	authenticate	certificate
connection	cache	bypass	client
credential	function	cause	code
crossite	host	change	command

Table 5 shows that there will be a tendency for word redundancy or similar meanings, e.g code and command in TargetImpact. This issue is taking care by constructing the similarity list discussed in the next subsection.

C. Domain Specific Wordlist (DSWordlist)

Domain specific wordlist (DSWordlist) is the phase where the words are groups not only based on the grammar usage of singular or plural, or tenses, but also based of their semantic domain understanding, for example, an email and a mail , both should be considered as in one group of word. First, using one-to-one matching, the definitions of the words are listed as shown in the sample in Table 6.

TABLE VI. WORDS, DEFINITION AND CLASSIFIERS

Words	Definition	Classifier
setting	change qualities of how the application works.	SourceRoot
configuration	is often where an application is customized for user or group	SourceRoot
function	a function returns a value	SourceLocation
parameter	are the strings/arguments used to pass value to functions or programs	SourceLocation
hijack	to stop and steal	TargetVector
spoof	to imitate	TargetVector
database	place where records stores in it	TargetImpact
script	written code for run-time environment	TargetImpact

Table 7 shows the words, their meanings and the classifiers they belong to. However, if one search for the incident is using function and another is using program, there is chance of ambiguity to locate the patterns, as these two words may refer to the same object.

Therefore, the second process took place. Second process, the domain specific word list (DSWordlist) is defined as the list of words that are referring to the domain semantic meanings in their specific classifier. An example of (DSWordlist) is given in Table 7.

TABLE VII. DOMAIN SPECIFIC WORD LIST WITH SEMANTIC

Wordlist	Semantic Meanings	Classifier
connection request session	activity between two machines	SourceRoot
function program system	workable code to perform instruction	SourceLocation
denial block terminate crash	ability to deny	TargetVector
account system password	user belonging identity	TargetImpact

Table 7 shows the semantic meaning for the words. The first column is the words from the data set. The second column is the meaning and the third column is the classifier they belong to. In this example, the function and program has been defined as in similar group of object. The wordlist and their classifiers are the input to the next phase: domain specific schema grammar (DSSchema).

D. Domain Specific Schema Grammar (DSSchema)

Domain specific schema grammar is the general description how the classifiers worked and used to execute the classification. In this study, the schema is represents using Backus- Naur Form (BNF) notation. The words in the sentence are referred as field. The target of this classification is to define the pattern. Each pattern is characterized by the four classifiers. The classifiers are SourceRoot, SourceLocation, TargetVector and TargetImpact. In this study, the process begins with inquiry of the incidents existed in the file. Each incidents consists of incidents name (the CVE number) and patterns (the sentence),

<incidents> = <incidentsname> + <patterns>

Upon this phase, the inquiry will focus on patterns as target. At this point, the DSMarker (the primary marker) is use to identify between the source and target.

<patterns> = <source> |< primarymarker> | <target>
<target> = <targetvector> | <subsequentmarker> |
<targetimpact>

At this point, the inquiry will focus on the first classifier, the sourceroot, to investigate the domain specific wordlist (DSSWordlist) within this classifier. Follow by the next classifiers and their DSSWordlist. The whole process is given as,

<patterns> = <sourceroot>| <subsequentmarker>|
< sourcelocation>|<primarymarker>|
<targetvector>|<subsequentmarker>|
< targetimpact>

This process produced the generative grammar and formed a new notation refer as vulnerability flow diagram as in [38] and later used in the new framework and refer as vulnerability classification pattern or VulClaP.

E. Accepting Affirmation

Accepting affirmation phase is one of most important steps during the entire process, whose task is to perform the validation for the generic vulnerability classification framework.

Vulnerability Classification Pattern Algorithm. The purpose of the algorithm is to support the generic vulnerability classification framework. It should be validated to check its accuracy.

Vulnerability Classification Framework. This framework served as the final product, therefore an affirmation to accept it is important and execute in this phase.

Experimental Result. This result come from the quantitative approach to show the accuracy by using the precision and recall rate. Five random datasets were used and each consist 500 records.

Validation Result. This result comes from the qualitative approach to show the acceptance of the experts by using a questionnaire. The experimental result also was supported with feedback from the experts.

VI. DATA COLLECTION

This study intends to observe the vulnerability pattern from a reported threat or attack incidents. These incidents are collected and monitored by few organizations such as NVD, CVSS, and CWE (Common Vulnerabilities and Exposures: The Standard for Information Security Vulnerability Names). Used as the case study, these reports were assessed and debated by the experts in the field.

In order to study the pattern of the reported incident, a sample is needed based on certain stratum. The target population is aim within the first five years of the reports. In this case, the random sampling is used. The size shall determine the generality of the results and the ability to detect true effects. The data collection must also adhere to the conditions as in Table 8.

TABLE VIII. DATA CHARACTERISTICS

Characteristics	Description
Reported incidents	The dataset must be reported of original incidents
Endorsed by authorized organization	The data must be reported and verified by an authorized security organization
Open to public	The data is open for public as reference
Sample size must be more than 500 data	The quantity of sample in a data must be able to represent all possible patterns

Table 8 shows that the data to be used in this study must be reported incidents, endorsed by authorized organization which having their own of third party security auditor, and should be open to public as it indicates a collective and ability to produce a predictive trend [39]. In market, the vulnerability database can be from local authorities or international authorities. The local authorities are varies and comes from both profit-based organisation and nonprofit-based organisation. However, the samples available are limited. Thus, this study turns to databases maintained by international authorities. There are two major open vulnerability databases that have international authorities:

- Open Source Vulnerability Database (OSVDB)
- National Vulnerability Database (NVD)

The OSVDB is an open to public, independent vulnerability database which founded in 2002. However, recently, in April 2016 this database had been closed as announced in an article by Jon Gold at www.networkworld.com. Hence, this research used the reported vulnerability database of Common Vulnerability Exposure or also known as CVE, as uses by numbers or researchers including [40],[41],[3] and [42]. Even though, there are some inconsistency issues in CVE as mentioned in their website, it is treat as minor and has been overcome during the implementation process [42]. In addition, the database is being used in security research such as [43] and [44] due to its ability to produce trends in threat or vulnerability. Nevertheless, the database also comprised a number of security organizations who found, reported and confirmed the cases such as, SANS and they claimed that despite the issues, the CVE has been used as a de facto standard in security industry [45].

A. Threat Model versus CVE

According to the current CVE website, the data introduce 13 categories of vulnerabilities. They are: Denial of Service (DoS), Code Execution, Overflow, Memory Corruption, Sql Injection, XSS, Directory Traversal, Http Response Splitting, Bypass something,Gain Information, Gain Privileges, CSRF and File Inclusion. An example of listing from year 1999 to 2008 is given in Fig. 6.

Vulnerabilities By Type

Year	# of Vulnerabilities	DoS	Code Execution	Overflow	Memory Corruption	Sql Injection	XSS	Directory Traversal	Http Response Splitting	Bypass something	Gain Information	Gain Privileges	CSRF	File Inclusion
1999	894	177	112	172			2	7		25	16	103		
2000	1020	257	208	206		2	4	20		48	19	139		
2001	1677	403	403	297		7	34	123		83	36	220		2
2002	2156	498	553	435	2	41	200	103		127	74	199	2	14
2003	1527	381	477	371	2	49	129	60	1	62	69	144		16
2004	2451	580	614	410	3	148	291	110	12	145	96	134	5	38
2005	4935	838	1627	657	21	604	786	202	15	289	261	221	11	100
2006	6610	893	2719	663	91	967	1302	322	8	267	271	184	18	849
2007	6520	1101	2601	953	95	706	884	339	14	267	323	242	69	700
2008	5632	894	2310	699	128	1101	807	363	7	288	270	188	83	170

Fig. 6. CVE Vulnerability Type - An Example from 1999-2008 (source: CVE website)

Fig. 6 shows the vulnerability type from MITRE CVE website. However, it was admitted in their FAQ page that this preliminary classification and categorization of vulnerabilities were too rough to be used to identify and categorize the functionality offered. Therefore, a mapping to a threat model is needed to fine grain the categories. As mentioned in section 2.5 at Table 2 and comparison in [27], threat model are varies, and the STRIDE model is light but comprehensive and less likely to be extended but convinced enough to be used in major model threat cases - a reasonable fact to choose this model as mentioned in [46] and [47]. The mapping from this category to STRIDE model is given in Table 9.

TABLE IX. MAPPING CVE TYPE TO STRIDE MODEL

STRIDE	CVE Threat Type
Spoofing	Http Response Splitting, Sql Injection, XSS
Tampering	Memory Corruption, File Inclusion
Repudiation	Bypass something, CSRF
Information Disclosure	Gain Information, Directory Traversal
Denial of Service	DoS, Overflow
Elevation of privilege	Gain privilege, Code Execution

Table 9 shows the threat type as defined in CVE website which made it maps to the threat model of STRIDE. This mapping is used as threat type total when random selection of data was made to allow VulClaP execute on it. Meaning, for any randomly selected CVE record, it will be checked on what threat type that they roughly been categorized in before compare with the execution result for that record.

VII. RESULT

This section explains the results discussion of experimental result and validation result.

A. Experimental Output

The analysis was conducted on three datasets: DS1, DS2 and DS3. Each dataset contains 500 data. Data set DS1 consist the data from year 1991-2004. Data set DS2 consist data from year 2005-2008 and dataset DS3 contains data of 2009-2016. All data were randomly selected based on a simple random function. The CVE Threat Type for all of them were checked by referring the CVE number. Next, they are put into the STRIDE class. The details for each dataset are given in Table 10.

TABLE X. DATA SET FROM CVE – DS1, DS2 AND DS3

From CVE Threat Type to STRIDE	DS1	DS2	DS3
Spoofing	10	10	8
Tampering	58	145	238
Repudiation	105	43	19
Information Disclosure	38	74	35
DDoS	96	118	98
EoP	123	89	85
Not classified	70	21	17
Total	500	500	500

Table 10 listed the numbers of patterns that been successfully gained from CVE database after mapping their category with STRIDE. Those that not detected, is categorized as Not classified. For example, in CVE-2002-1932: Microsoft Windows XP and Windows 2000, when configured to send administrative alerts and the 'Do not overwrite events (clear log manually)'option is set, does not notify the administrator when the log reaches its maximum size, which allows local users and remote attackers to avoid detection. This incident was not able to be classified by the algorithm therefore considered as Not classified. The algorithm had successfully detected the words that associate with attackers, analyzed them but however, could not mapped them to associate with TargetImpact category.

The rest of this subsection presents the execution of the vulnerability pattern algorithm into the CVE datasets. The execution was conducted on two modes: a) the mix-based datasets and b) the vendor-based datasets. The mix-based datasets refer to the random selection of data with different range of years. The vendor-based datasets refer to the difference of problems reported by either hardware or software vendors. For the purpose of this paper, we present the discussion on the mix-data set as the space constraints. The analysis was conducted on three datasets: DS1, DS2 and DS3. Each dataset contains 500 data. Data set DS1 consist the data from year 1991-2004. Data set DS2 consist data from year 2005-2008 and dataset DS3 contains data of 2009-2016.

The details for each dataset are given in Table 11. Table 11 listed the numbers of patterns that been successfully gained from each dataset by using the vulnerability classification for patterns (VulClaP) algorithm – patterns are classified into six classes with another one class that is Not classified. Those that not detected, is categorized as Not classified. For example, in CVE-2002-1932: Microsoft Windows XP and Windows 2000, when configured to send administrative alerts and the 'Do not

overwrite events (clear log manually)'option is set, does not notify the administrator when the log reaches its maximum size, which allows local users and remote attackers to avoid detection. This incident was not able to be classified by the algorithm therefore considered as Not classified. The algorithm had successfully detected the words that associate with attackers, analyzed them but however, could not mapped the avoid and detection to associate with TargetImpact category. The accuracy of the model is measured using two metrics: precision and recall rate.

Precision is the degree of confidence that the returned patterns are accurate when the VulClaP algorithm is applied on the data set. Recall is the degree of the ability to return the patterns when the VulClaP algorithm is applied on the data set. As given in the definition, between the two rates, the precision rate suggests a better understanding of accuracy in the model. Therefore, in later discussion the accuracy rate is used interchangeably with the precision rate. For the purpose of this paper, the discussion will use the first data set only, DS1. The data has been analyzed and the summary is given in Table 11.

Table 11 shows the precision and recall rate for DS1 data set. This table informs about how good or bad the prediction could be made from the 500 samples of incidents that range from year 1999 to 2004. The left column is the number of actual pattern classes from the data. The middle column is the number of true selected pattern that gained when analyzes using the VulClap. The aim of the analysis is to predict the value of the pattern based on several input of classifiers, which, in this research are the four classifiers SourceRoot, SourceLocation, TargetVector and TargetImpact. And the right most columns are the total counts of the predicted pattern and their precision rate, or accuracy.

TABLE XI. THE PRECISION AND RECALL RATE FOR DS1 DATA SET

Actual Pattern	**True Selected Pattern**							**Total**	**P Rate**
	S	**T**	**R**	**I**	**D**	**E**	**NC**		
S	8	0	0	0	0	0	2	10	0.80
T	1	53	1	1	1	0	1	58	0.91
R	0	2	100	0	0	3	0	105	0.95
I	0	1	1	34	0	2	0	38	0.89
D	0	2	0	0	93	0	1	96	0.97
E	0	1	3	2	0	116	1	123	0.94
NC	1	1	0	0	0	1	67	70	0.96
Total	10	60	105	37	94	122	72	500	0.92
R Rate	0.80	0.88	0.95	**0.92**	0.99	0.95	0.93		0.92

The last rows are the total counts of the recalled patterns and their recall rates. The last cell diagonally at bottom right denotes the sample size from either the sum total from row of precision or recall .The bold values are the average rate for the precision and recall rates. In this data set, the average rate for precision and recall are both 0.92. The shaded areas are the count of true predicted pattern in proportion of the total recall or precision. Or, reading top-down, the analysis also shows that VulClaP algorithm had recalled 105 samples to be in class Repudiation, denotes by the last total row. And from that total, 100 samples are truly recalled as the Repudiation class, and another five samples have been falsely recalled as EOP (3 samples), Tampering (1 sample) and Information Disclosure (1 sample), make the recall rate as 0.92.

B. Expert Validation

The validation was also done through expert opinion to verify the correctness of the classification categories. The experts are the people who is working in academic or security industry with more than five years of experience, and may have additional related professional certificate on top of their knowledge and job experience. Five experts were selected to argue on the vulnerability categories and three experts validate on the frameworks pragmatic. The results justified that issues such as ambiguity of words, redundancy in category, the influenced of certain conjunction such as 'and' , 'or', the generality of certain terms regardless of the application version, the used of newer words are still able to be classified its respected class. In addition, the slight changes of CVE sentence structure and the usage of uncommon word were also highlighted in the expert questionnaire and resolved as less significant.

VIII. CONCLUSION

After the proposed work on the formal syntax and semantic by using the BNF and the relationship notation, two activities under analysis process are enhanced. The enhanced framework is given in Fig. 7.

The red activities show the enhanced part. The issue of high potential in redundancy is resolved by using a formal BNF to represent the syntax and semantic schema. And the relationship of vulnerability classifier is representing using the vulnerability flow diagram.

In particular, this study contributes to the software industry by assisting to formulate a secured software process framework with vulnerability classification algorithm and vulnerability flow diagram - an area that has been underestimating by researches due to the difficulty to generalize the outcomes. The significant of this secured framework that makes it different from others is the ability of the vulnerability flow diagram to visualize the precedent and antecedent of an exploit using the simplified vulnerability pattern.

The main challenging task in this study is to formulate the domain specific wordlist and domain specific schema for the classification which is proposed to be further look into in future work.

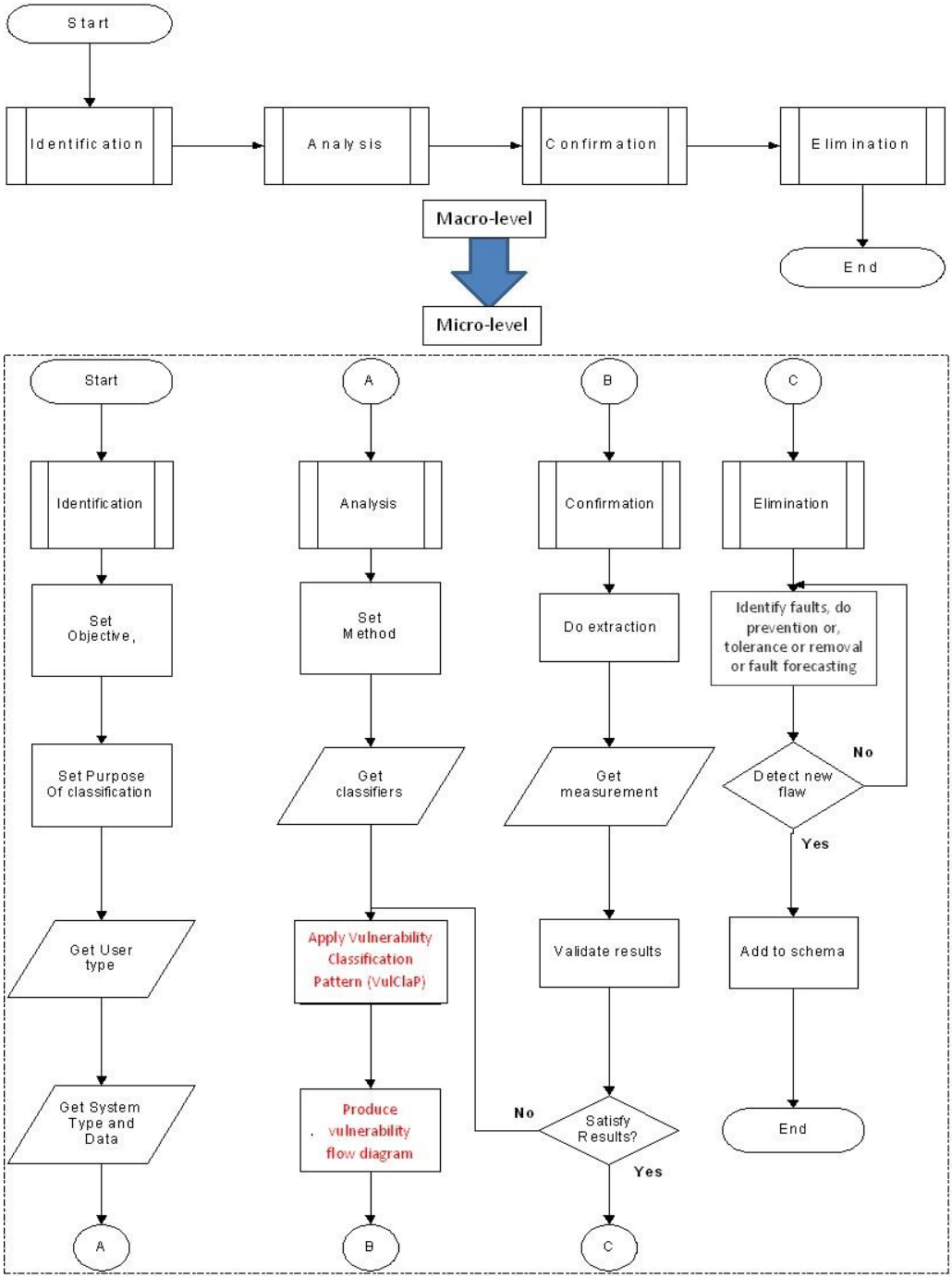

Fig. 7. The Enhanced Vulnerability Classification Framework (E-VulCaF)

ACKNOWLEDGMENT

This project is supported by grant of PJP/2018/FTMK(4B)/S01631 from Center For Research and Innovation Management (CRIM), UTeM.

REFERENCES

[1] O. Rebolloa, D. Melladob, E. Fernandez-Medinac, and H. Mouratidis, "Empirical Evaluation Of A Cloud Computing Information Security Governance", Information and Software Technology, vol. 58(3): pp. 44–57, 2015.

[2] R. Shaikh and M. Sasikumar, "Data classification for achieving security in cloud computing", Procedia Computer Science International Conference on Advanced Computing Technologies and Applications (ICACTA), vol.45, pp. 493 – 498, 2015.

[3] J. Ruohonen, S. Hyrynsalmi, S. Rauti, and V. Leppänen, "Mining Social Networks Of Open Source Cve Coordination", Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement, IWSM Mensura '17, ACM, New York, NY, USA, pp. 176–188, 2017.

[4] A. Tripathi and U. Singh, "Taxonomic Analysis Of Classification Schemes In Vulnerability Databases", Computer Sciences and Convergence Information Technology (ICCIT), 6th International Conference on, pp. 686–691, 2011.

[5] E.W. Burger, M. D. Goodman, P. Kampanakis, and K.A. Zhu, "Taxonomy Model For Cyber Threat Intelligence Information Exchange Technologies", Proceedings of the ACM Workshop on Information Sharing, vol.38, pp. 51–60, 2014.

[6] R. Abbott, J. Chin, J. Donnelley, W. Konigsford, S. Takubo and D. Webb, Security Analysis and Enhancement of Computer Operating Systems, NBSIR pp.76-1041, 1976.

[7] R. Bisbey, and D. Hollingworth. "Protection Analysis: Final Report", Technical Report ISI/SR-78-13, Technical report, Information Sciences Institute, University of Southern California, 1978.

[8] E. L.Carl, R.B. Alan, P.M. John, and S.C. William, "A Taxonomy of Computer Program Security Flaws", ACM Computer Survey vol.26, pp. 211–254, 1994.

[9] Aslam, T., Krsul, I. and Spafford, E., "Use of A Taxonomy of Security Faults", 19th National Information Systems Security Conference, 1996.

[10] J. C. Munson, A.P. Nikora, and J.S Sherif, "Software Faults: A Quantifiable Definition", Advances in Engineering Software vol.35, pp.327–333, 2006.

[11] G. S. Walia, and J.C. Carver, "A Systematic Literature Review to Identify and Classify Software Requirement Errors", Information and Software Technology vol.51 pp. 1087–1109, 2009.

[12] J. Luo, K. Lo, and H.Qu, "A software vulnerability rating approach based on the vulnerability database", pp 1–9, 2014.

[13] M. Bishop, "Classifying Vulnerabilities", 1999.

[14] C. Weissman, "System Security Analysis Certification Methodology and Results", Technical Report SP-3728, Technical report, System Development Corporation, 1973.

[15] D.L. Lough, Thesis: A Taxonomy of Computer Attacks with Application to Wireless Network, Technical report, Virginia Polythechnic Institute, 2001.

[16] R.C. Seacord, and A.D. Householder, "A Structured Approach to Classifying Security Vulnerabilities", CMU/SEI-2005-TN-003, Technical report, Carnegie Mellon, Software Engineering Institute, 2005.

[17] S.Eagle, S.Whalen, D. Howard, and M. Bishop, "Tree approach to vulnerability classification", Technical Report CSE-2006-10, Technical report, Department of Computer Science, UC Davis, 2006.

[18] B. Biggio, , G. Fumera, and F. Roli, "Security Evaluation Of Pattern Classifiers Under Attack", IEEE Trans. Knowl. Data Eng. vol.36, pp. 984–996, 2014.

[19] A. Bollin, "Crossing the borderline- from formal to semi formal specifications", IFIP of Software Engineering Techniques: Design For Quality, Vol. 227, pp. 73–84, 2006.

[20] L. Lowis, and R. Accorsi, "Vulnerability Analysis In Soa Based Business Processes", IEEE Transaction on Services Computing, vol.4, pp.230–242, 2011.

[21] D. Last, "Using Historical Software Vulnerability Data To Forecast Future Vulnerabilities", IEEExplore, 2015.

[22] C. Haley, R. Laney, J. Moffett, J. and B.Nuseibeh, "Security Requirements Engineering:A Framework For Representation And Analysis", IEEE Transactions on Software Engineering, vol.34, pp.133–153, 2008.

[23] Lamsweerde, A. v., Brohez, S., Landtsheer, R. D. and Janssens, D., 2003. From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering, Requirements for High Assurance Systems (RHAS'03), pp. 49–56.

[24] R. Villarroel, E. Fernandez-Medina, and M. Piattini, "Secure Information SystemsDevelopment - a Survey and Comparison", Computers & Security ,vol.24, pp.308–321, 2005.

[25] V.M. Igure and D.R. Williams, "Taxonomies of Attacks and Vulnerabilities in Computer Systems", IEEE Communication Surveys & Tutorials, vol. 10, pp 14, 2008.

[26] J. Maatta, J. Harkonen, T.Jokinen, M. Mottonen, P. Belt, M. Muhos, and H. Haapasalo, "Managing Testing Activities in Telecommunications: A Case Study", J. Eng. Technol.Manage. vol. 26: pp.24, 2009.

[27] N.H.Hassan, S.R. Selamat , S.Sahib and B.Hussin (2011) "Towards Incorporation of Software Security Testing Framework in Software Development". In: Mohamad Zain J., Wan Mohd W.M.., El-Qawasmeh E. (eds) Software Engineering and Computer Systems. ICSECS 2011. Communications in Computer and Information Science, Springer, Berlin, Heidelberg, vol 179, pp.16-30, 2011.

[28] A.Schipper, H.Fuhrmann, and R.V. Hanxleden, "Visual comparison of graphical models", vol.10, pp.335–340, 2009.

[29] F.A. Braz, E.B. Fernandez, E. B. and M.VanHilst, "Eliciting Security Requirements through Misuse Activities", 19th International Conference on Database and Expert Systems Application, pp. 328–333, 2008.

[30] M. Gupta, J. Walp, and R. Sharman, R., "Threats, Countermeasures and Advances in Applied Information Security", IGI Global, 2012.

[31] R.R. Linde, "Operating System Penetration", National Computer Conference, System Development Corporation, pp.8, 1975.

[32] S. Hansman, and R. Hunt, "A Taxonomy Of Network And Computer Attacks", Computers and Security vol.24, pp. 31 – 43, 2005.

[33] T. Katrina, C. Brian, and M. Gary, "Seven pernicious kingdoms: a taxonomy of software security errors", IEEE Security and Privacy, vol. 3, pp.81–84, 2005.

[34] K. Jiwnani, and M. Zelkowitz, "Maintaining Software with a Security Perspective", Proceedings of International Conference on Software Maintenance, pp. 194–203, 2002.

[35] B.Potter, and G. McGraw, "Software Security Testing", Security & Privacy, IEEE vol.2,pp. 81–85, 2004.

[36] L.Desmet, P. Verbaeten, W. Joosen, and F. Piessens, "Provable Protection against Web Application Vulnerabilities Related to Session Data Dependencies", IEEE Transactions on Software Engineering, vol.34, pp. 50–64, 2008.

[37] A.M. Memon, " A comprehensive framework for testing graphical user interfaces",2001.

[38] N.H.Hassan, S.R. Selamat and S.Sahib, "Establishing the Relationship in Vulnerability Classification for a Secure Software Testing", International Conference on Advances in Intelligent Systems in Bioinformatics, Chem-Informatics, Business Intelligence, Social Media and Cybernetics (IntelSys), 2014.

[39] K. Alzhrani, E.M. Ruddy, T.E. Boulty, and C.E. Chow, "Automated Big Text Security Classification", IEEE Intelligence and Security Informatics, pp. 1–6, 2016.

[40] Z. Moghbel, and N. Modiri, "A Framework For Identifying Software Vulnerabilities Within Sdlc Phases", International Journal of Computer Science and Information Security, vol.9, pp. 203, 2011.

[41] U.K. Singh, and C. Joshi, "Network security risk level estimation tool for information security measure", IEEE 7th Power India International Conference (PIICON), pp. 1–6, 2016.

[42] D.Papp, Z. Ma, and L. Buttyan, "Embedded Systems Security: Threats, Vulnerabilities, And Attack Taxonomy", 13th Annual Conference on Privacy, Security and Trust, Institute of Electrical and Electronics Engineers Inc., pp. 145–152, 2015.

[43] S.Neuhaus and T.Zimmermann, "Security Trend Analysis With CVE Topic Models", IEEE 21st International Symposium on Software Reliability Engineering (ISSRE), pp. 111–120, 2010.

[44] A.Barua, S.W. Thomas, and A.E. Hassan, "What are developers talking about? An analysis of topics and trends in stack overflow", Empirical Software Engineering, vol.19, pp. 619–654, 2014.

[45] J.Connolly, M. Davidson, M. Richard, and C. Skorupka, "The Trusted Automated Exchange Of Indicator Information", MITRE Corporation, pp. 1–20, 2012.

[46] R. Khan, K. McLaughlin, D. Laverty, D. and S. Sezer, "STRIDE-based Threat Modeling for Cyber-Physical Systems", IEEE, 2018.

[47] M. Ostkamp, C. Kray, and G. Bauer, "Towards A Privacy Threat Model For Public Displays", Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems, EICS ACM, pp. 286–291, 2015.