УДК 004.93`14

ПОДХОДЫ К АВТОМАТИЗАЦИИ ПРОЦЕССА ВАЛИДАЦИИ УЯЗВИМОСТЕЙ, НАЙДЕННЫХ АВТОМАТИЧЕСКИМИ СКАНЕРАМИ БЕЗОПАСНОСТИ, ПРИ ПОМОЩИ НЕЧЁТКИХ МНОЖЕСТВ И НЕЙРОННЫХ СЕТЕЙ

[1]Гильмуллин Т.М., [2]Гильмуллин М.Ф.

[1]ЗАО «Позитив Текнолоджиз», Москва, e-mail: tim55667757@gmail.com;
[2]Елабужский институт (филиал) ФГАОУ ВПО «Казанский (Приволжский) федеральный университет», Елабуга, e-mail: Gilmullin.Mansur@gmail.com

В статье рассмотрены и формально решены проблемы автоматической классификации уязвимостей информационных систем. Отмечена сложность составления актуального списка кандидатов в уязвимости веб-приложений. Поставлена задача нечёткой классификации уязвимостей. Проанализированы возможные способы решения данной задачи, среди которых особо выделен математический аппарат нейронных сетей. Выбраны и построены измерительные шкалы для чёткой и нечёткой оценки свойств уязвимостей и степеней их принадлежности классам. Для различных интерпретаций результатов и связи между шкалами указаны функции фазификации и дефазификации. Предложена матрица кодирования свойств уязвимостей для подготовки обучающих векторов-признаков на входы нейронной сети. Предложена архитектура нейронной сети для классификации уязвимостей. Разработаны программные модули FuzzyClassificator для нечёткой классификации произвольных объектов, представленных векторами признаков для различного числа классов и структуры нейронной сети. Процесс работы программы представлен в виде функциональной IDEF0-модели. Приведен пример использования программы: подготовка входных данных, обучение, классификация, анализ результатов.

Ключевые слова: нейронная сеть, программные модули, вектор признаков, классификация, нечеткие шкалы, разделение элементов, уязвимости, информационная безопасность, обучение нейросети

APPROACHES TO AUTOMATE VALIDATION OF VULNERABILITIES FOUND BY AUTOMATIC SECURITY SCANNERS USING FUZZY SETS AND NEURAL NETWORKS

[1]Gilmullin T.M., [2]Gilmullin M.F.

[1]CJSC Positive Technologies, Moscow, e-mail: tim55667757@gmail.com;
[2]Elabuga Institute (branch) of Kazan (Volga Region) Federal University, Elabuga, e-mail: Gilmullin.Mansur@gmail.com

The article discusses and formally solves the problem of automatic classification of information system vulnerabilities. The complexity of compiling an up-to-date list of potential web application vulnerabilities is shown. The problem of fuzzy classification of vulnerabilities is set. Practicable methods of problem solution are analyzed; the mathematical formalism of neural networks is considered most closely. Measuring scales for classic and fuzzy evaluation of vulnerability properties and degree of vulnerability membership in classes are chosen and developed. For result interpretation and scale matching, fuzzification and defuzzification functions are given. A matrix for encoding vulnerability properties is proposed to prepare training attribute vectors for neural network input. A neural network architecture is suggested to classify vulnerabilities. FuzzyClassificator software modules are developed for fuzzy classification of arbitrary objects represented with attribute vectors for different neural network structures and numbers of classes. The program operation process is shown as a functional IDEF0 model. The article contains an example of program application including preparation of input data, training, classification, and output analysis.

Keywords: neural networks, software modules, a feature vector, classification, fuzzy scale, separation of elements, vulnerability, information security, training the neural network

В настоящее время существует огромное количество различных **сканеров информационной безопасности** (далее – сканеры), применяемых для анализа защищенности веб-приложений: Acunetix WVS, IBM AppScan, BurpSuite, NMap, HP Fortify, Positive Technologies – MaxPatrol, XSpider, Application Inspector и других, различающихся ценой, качеством сканирования, поддерживаемыми технологиями, типами обнаруживаемых уязвимостей, методиками поиска – белый или черный ящик, и десятками других параметров. Некоторое представление о возможностях сканеров и сравнение их характеристик можно получить, например, из периодического отчета «The Web Application Vulnerability Scanners Benchmark (by Shay Chen, Information Security Researcher)» [6].

При разработке сканеров информационной безопасности важную роль играют методики тестирования их работы, о которых мы упоминали в статье «Тестирование сканеров безопасности веб-приложений: подходы и критерии» [5]. В этих методиках

особое место занимает конкурентный анализ сканеров и их сравнение с другими.

Основным ожидаемым результатом работы любого сканера безопасности является **список кандидатов в уязвимости**, полученный в процессе анализа веб-приложения. Использование в сканерах сложных эвристических алгоритмов часто приводит к большому числу ложных срабатываний и заполнению такого списка несуществующими в реальном веб-приложении уязвимостями (false positives). В связи с чем требуется длительная работа экспертов-аналитиков в области информационной безопасности, чтобы перепроверить, подтвердить или опровергнуть найденные автоматическим сканером кандидаты в уязвимости. При этом они используют как различные инструментальные средства для попыток эксплуатации уязвимостей, так и делают выводы, опираясь на собственный опыт.

На практике, как один из способов подтверждения уязвимости, может использоваться сравнение с некоторыми аналогичными уязвимостями, о которых заранее известно, что они имеются в похожем веб-приложении или в том же веб-приложении, но более ранней версии. Для этого может быть сформирована база данных «эталонных» уязвимостей, содержащая характеристики и описание реальных, ранее найденных уязвимостей. Тогда эксперт-аналитик сможет сделать дополнительные выводы и подтвердить или опровергнуть новые кандидаты в уязвимости, опираясь на их сравнение с эталонами по некоторым правилам. Однако подобная рутинная работа также нуждается в дополнительных инструментах и процедурах, позволяющих выполнить такой анализ: сравнивать кандидаты с эталонными уязвимостями и отсеивать очевидные false positives.

Постановка задачи нечёткой классификации уязвимостей

Как отмечено выше, проблема подтверждения уязвимостей из списка кандидатов на практике может решаться как задача сравнения их с некоторыми эталонами. В случае если все объекты – и эталоны, и кандидаты в уязвимости – могут быть однозначно параметризованы, представлены в виде вектора признаков объекта, то проблема может быть сведена к классической задаче классификации элементов множества [4, п. 3].

Входные данные:

1. Задано n-мерное векторное пространство ***Vulners*** – всех уязвимостей (vulnerabilities) веб-приложений, которые могут быть заданы векторами признаками v_i:

$$Vulners = \left\{ v_i \in R^n \mid v_i = (v_1, \ ..., v_n),\ n = \text{const} \right\},$$

где v_1, ..., v_n – чёткие, либо дефазифицированные из нечётких, числовые характеристики отдельных признаков уязвимости.

Из *Vulners* выделено непустое конечное подмножество ***Candidates*** – кандидатов в уязвимости для некоторого веб-приложения:

$$Candidates \neq \varnothing,\ Candidates \subset Vulners,$$

$$Candidates = \left\{ c_i \mid c_i \in Vulners,\ \ i = \overline{1,m},\ \ m = \text{const} \right\}.$$

Элементы из *Candidates* являются объектами классификации.

2. Каждую уязвимость из *Candidates* допускается отнести к двум классам:

– **I класс** подтвержденных (verified) уязвимостей:

$$Ver \subseteq Candidates,$$

$$Ver = \left\{ ver_i \mid ver_i \in Candidates,\ i \le m \right\},$$

– **II класс** неподтвержденных (non-verified) уязвимостей:

$$NVer \subset Candidates,$$

$$NVer = \left\{ nver_i \mid nver_i \in Candidates,\ i < m \right\}.$$

3. Существует ***Eth*** – непустое подмножество «эталонных» (ethalon) уязвимостей, входящих в I класс:

$$\exists Eth \neq 0,\ Eth \subseteq NVer,$$

$$Eth = \left\{ e_i \mid e_i \in Ver,\ \ i = \overline{1,k},\ k = const,\ \ k \le m \right\}.$$

Используя элементы *Eth*, допускается делать предположения о принадлежности других элементов из *Candidates* классам *Ver* и *NVer* в виде некоторого «правила» – функции f от двух переменных, определенной на множестве $Ver \times Eth$ со значениями в *Ver*, то есть выполняется условие:

$$Ver \cup NVer = Candidates,$$

$$v \in Ver \Rightarrow \exists f\ \exists E \subseteq Eth\ \ f(v, E) \in Ver.$$

4. Задано множество измерительных шкал ***Scales*** для оценки чётких и нечётких характеристик уязвимостей:

$$Scales = \{S_p, S_f\},$$

где S_p – чёткая (precise) числовая шкала; S_f – нечёткая (fuzzy) шкала лингвистических переменных.

Схематично условия задачи представлены на рис. 1.

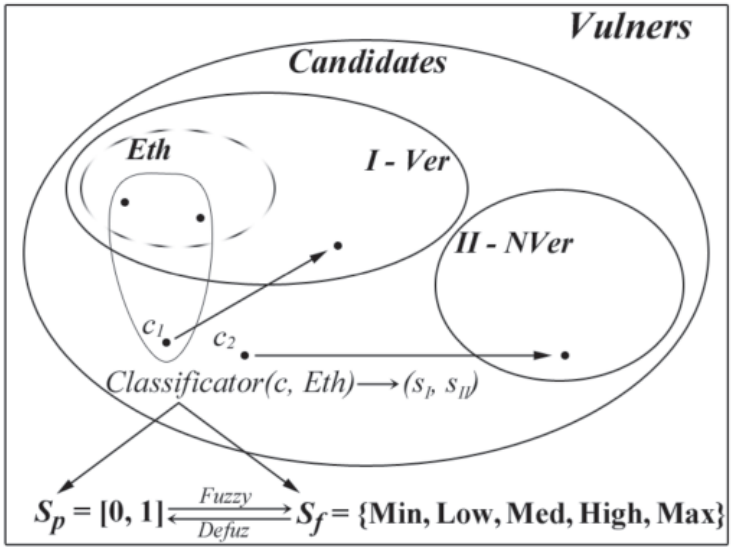

Рис. 1. Диаграмма Эйлера – Венна для задачи классификации уязвимостей

Требуется:

1. Построить отображения ***Fuzzy*** и ***Defuz*** для задачи интерпретации результатов классификации и оценок:

$Fuzzy : S_p \rightarrow S_f$;

$Defuz\ S_f \rightarrow S_p$; $S_p, S_f \in Scales$,

определяющие способы представления чётких оценок в виде нечёткого значения некоторой лингвистической переменной и наоборот.

2. Построить отображение ***Classificator*** для задачи классификации:

$$Classificator : Candidates \times Eth \rightarrow Scales \times Scales,$$

ставящее в соответствие каждой уязвимости из *Candidates* оценки принадлежности уязвимости каждому классу, учитывая при этом имеющиеся эталоны. Иначе говоря, нужно построить функцию

$$Classificator(c, Eth) \rightarrow (s_I, s_{II}),\ c \in Candidates,\ s_I, s_{II} \in S_p \vee s_I, s_{II} \in S_f.$$

Измерительные шкалы и связь между ними

Среди исследований, выполненных в работе [3], приведены примеры, показывающие, что в качестве универсальных измерительных шкал для оценки свойств информационных систем могут быть использованы:

– множество действительных чисел из отрезка [0, 1] [3, с. 62–64], которое легко может быть преобразовано в любые другие виды чётких числовых множеств: дискретных, непрерывных, неограниченных, при помощи различных функций конвертирования [3, с. 66–67];

– *F*-множества упорядоченных нечётких переменных [3, с. 59–60] вида $FP = \{fp_i\}$, где fp_i – лингвистические переменные, описывающие значения свойств объекта.

Дадим некоторые определения из теории нечётких систем, связанные с данным исследованием.

Нечёткое множество *A* в полном пространстве *X* определяется через **функцию принадлежности** (membership function):

$$\mu_A : X \rightarrow [0, 1].$$

Величина $\mu_A(x)$, $x \in X$ интерпретируется как субъективная оценка степени принадлежности элемента *x* к нечёткому множеству *A*.

Носителем нечёткого множества или **несущим множеством** *A* называется чёткое подмножество полного пространства *X*, на котором значение $\mu_A(x)$ положительно:

$$\sigma(A) = \{x \in X \mid \mu_A(x) > 0\}.$$

Нечёткими *F*-множествами называют совокупность всех нечётких подмножеств $F(X)$ произвольного базового множества *X*, а их функции принадлежности – ***F*-функциями**. Как правило, под μ_A понимают сужение функции принадлежности со всего *X* на $\sigma(A)$, поэтому *F*-множества

обычно задают функцией принадлежности и несущим множеством:

$$A = \langle \mu_A(x), \sigma(A) \rangle.$$

Нечёткая переменная – это объект предметной области, характеризуемый тройкой $\{N, X, R(N, x)\}$, где N – название переменной, X – универсальное множество (полное пространство наблюдений) с базовой переменной x, $R(N, x) \in F(X)$ – нечёткое F-множество, задающее ограничения на значения переменной x, обусловленные её названием N [2].

Лингвистическая переменная – это нечёткая переменная, значениями которой являются другие нечёткие переменные: слова или предложения естественного или формального языка [2].

Нечёткая шкала (fuzzy scale) – это упорядоченная совокупность S нечётких переменных A_i, определенных своими F-функциями, значения из которой может принимать некоторая лингвистическая переменная s:

$$s \in S,$$

$$S = \left\{ A_i \mid A_i = < \left\langle \mu_{A_i}(x) , \sigma(A_i) \right\rangle, \ i = \overline{1,n} \right\}.$$

Фиксация значения $s = A_i$ для некоторого свойства означает, что оно оценивается лингвистической переменной s и имеет значение A_i. Для задания шкалы S необходимо определить все нечёткие переменные A_i, указать их функции принадлежности и несущие множества [3, с. 128].

Фактически нечёткая шкала является обобщением известного типа порядковых шкал, в которых элементы ранжированы и отражают качественную оценку некоторого свойства объекта. Количество уровней нечёткой шкалы и несущие множества для неё рекомендуется выбирать, опираясь на исследования функции и шкалы желательности Харрингтона [3, с. 125–126].

Функция желательности Харрингтона (desirable function) – это функция, которая отображает результаты оценок экспертов на отрезок [0, 1] и характеризует уровень «желательности» той или иной оценки [1]:

$$d(y) = e^{-e^{-y}},$$

или, в другой записи,

$$d(y) = \exp(-\exp(-y)).$$

Эта функция возникла в результате наблюдений за решениями экспертов о предпочтениях соотнесения результатов эксперимента со значениями на отрезке [0, 1]. Функция позволяет оценивать предпочтение оценок для объектов различной размерности и природы. Кроме того, в областях, близких к 0 и 1, её «чувствительность» существенно ниже, чем в средней зоне (см. рис. 2). Выбор значений 0,37 и 0,63 обусловлен удобством вычислений, так как $0{,}37 \approx e^{-1}$, а $0{,}63 \approx 1 - e^{-1}$. За начало отсчета обычно принимают значение в точке перегиба: $d(0) \approx 0{,}37$. Допускаются отклонения границ уровней шкалы желательности на ±0,03 [3, с. 100–101].

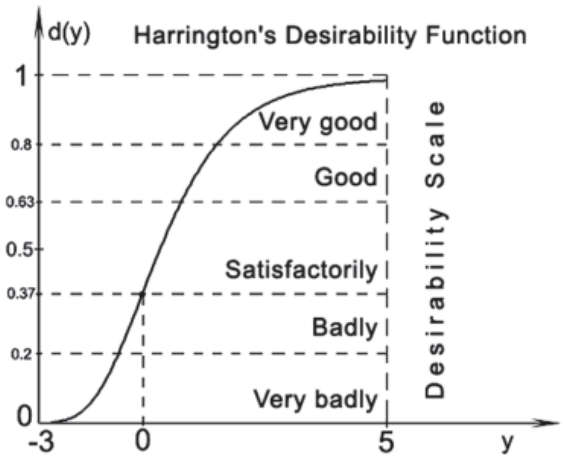

Рис. 2. Шкала желательности и функция желательности Харрингтона

Шкала желательности (desirability scale) – это психофизическая шкала, которая устанавливает соответствие между физическими параметрами свойств исследуемого объекта и психологическими, субъективными оценками экспертов «желательности» того или иного значения этих свойств [1].

Специальные функции принадлежности, используемые для построения нечётких шкал, задаются формулами [3, с. 198–202] и изображены на рис. 3.

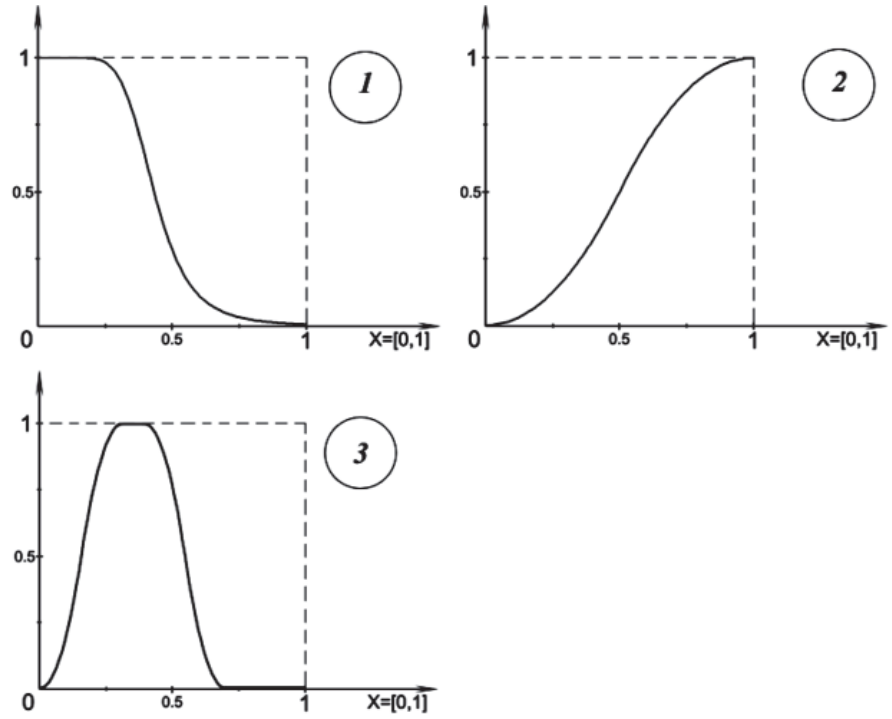

Рис. 3. Примеры специальных функций принадлежности, заданных своими параметрами: 1 – гиперболическая: $\mu_{hyperbolic}(x; 3; 5; 0,1)$; 2 – параболическая: $\mu_{parabolic}(x; 0; 1)$; 3 – колоколообразная: $\mu_{bell}(x; 0; 0,3; 0,4)$

1. **Гиперболическая**:

$$\mu_{hyperbolic}(x,a,b,c) = \begin{cases} 1, x \le c, \\ \dfrac{1}{1+(a(x-c))^{b}}, x > c. \end{cases}$$

2. **Параболическая**:

$$\mu_{parabolic}(x,a,b) = \begin{cases} 0, x \le a, \\ \dfrac{2(x-a)^2}{(b-a)^2}, a < x \le \dfrac{a+b}{2}, \\ 1-\dfrac{2(x-b)^2}{(b-a)^2}, \dfrac{a+b}{2} < x < b, \\ 1, x \ge b. \end{cases}$$

3. **Колоколообразная**:

$$\mu_{bell}(x,a,b,c) = \begin{cases} \mu_{parabolic}(x,a,b), x < b, \\ 1, b \le x \le c, \\ 1-\mu_{parabolic}(x,c,c+b-a), x > c. \end{cases}$$

Кроме аналитического способа функции принадлежности также могут быть заданы таблично, либо используя метод согласования мнений экспертов [3, с. 131–132].

Обобщая всё сказанное выше и опираясь на примеры шкал, приведенных в работе [3, с. 94–101], для задачи классификации уязвимостей предлагается использовать следующие шкалы из множества *Scale* (см. рис. 4):

1. **Для четкой шкалы** S_p выберем множество действительных чисел из отрезка [0, 1]:

$$S_p = \{s \mid s \in [0, 1]\}.$$

2. **Для нечёткой шкалы** S_f выберем универсальную шкалу лингвистических переменных

$$S_f = \{Min, Low, Med, High, Max\},$$

где лингвистические переменные задаются нечёткими множествами:

$$Min = \langle \mu_{Min}(x),\ \sigma(Min) \rangle;$$

$$\mu_{Min}(x) = \mu_{hyperbolic}(x,\ 8,\ 20,\ 0);\ \ x \in \sigma(Min) = [0;\ 0{,}23];$$

$$Low = \langle \mu_{Low}(x),\ \sigma(Low) \rangle;$$

$$\mu_{Low}(x) = \mu_{bell}(x;\ 0{,}17;\ 0{,}23;\ 0{,}34);\quad x \in \sigma(Low) = [0{,}17;\ 0{,}4];$$

$$Med = \langle \mu_{Med}(x),\ \sigma(Med) \rangle;$$

$$\mu_{Med}(x) = \mu_{bell}(x;\ 0{,}34;\ 0{,}4;\ 0{,}6);\quad x \in \sigma(Med) = [0{,}34;\ 0{,}66];$$

$$High = \langle \mu_{High}(x),\ \sigma(High) \rangle;$$

$$\mu_{High}(x) = \mu_{bell}(x;\ 0{,}6;\ 0{,}66;\ 0{,}77);\quad x \in \sigma(High) = [0{,}6;\ 0{,}83];$$

$$Max = \langle \mu_{Max}(x),\ \sigma(Max) \rangle;$$

$$\mu_{Max}(x) = \mu_{parabolic}(x;\ 0{,}77;\ 0{,}95);\quad x \in \sigma(Max) = [0{,}77;\ 1].$$

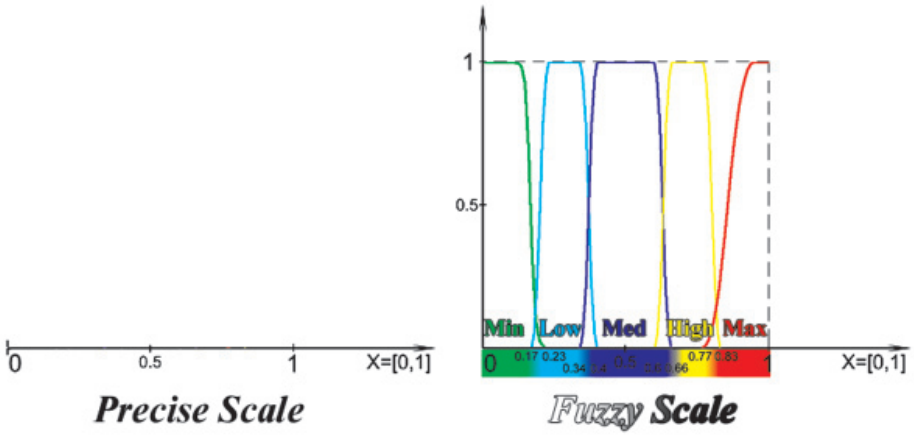

Рис. 4. Чёткие и нечёткие «универсальные» измерительные шкалы

В работе [3, с. 133–135] были определены и исследованы различные функции дефазификации и фазификации для решения практических задач.

Функцией дефазификации (defuzzification function) для нечёткого множества $A \in F(X)$, $X = [0, 1]$, заданного своей функцией принадлежности $\mu_A(x)$, называется любая функция $Defuz(\mu_A(x))$, возвращающая чёткое значение $x \in X$, «характерное» для A.

Функцией фазификации (fuzzification function) для чёткого значения переменной $x \in X$ и нечёткой шкалы $S = \{A_i\}$ называется любая функция $Fuzzy(x, S)$, возвращающая «наиболее подходящую» для x нечёткую переменную A_i.

Так как в нашем случае шкала S_p является множеством действительных чисел, а все функции принадлежности лингвистической переменной для шкалы S_f заданы аналитически, то в качестве функций связи между шкалами множества *Scales* выберем следующие:

$$Defuz(A_i) = \frac{\int_a^b x\mu_A(x)\,dx}{\int_a^b \mu_A(x)\,dx} \in S_p;\quad A_i \in S_f,\quad i = \overline{1,n};$$

$$Fuzzy(x,\ S_f) = A_i;\quad S_f = \{A_i\};\quad \forall j \neq i;\quad \mu_{A_j}(x) \le \mu_{A_i}(x),\quad i, j = \overline{1,n}.$$

Это означает, что для отображения *Defuz* чёткое значение нечёткого уровня A_i вычисляется по методу центра тяжести. А для отображения *Fuzzy* из двух смежных уровней A_i шкалы S_f выбирается та лингвистическая переменная, у которой значение функции принадлежности $\mu_{A_i}(x)$ наибольшее.

Кодирование входных данных

Для любого способа классификации уязвимостей они должны быть предварительно закодированы, то есть представлены вектором $v = \{v_i\} \in Vulners$. Для этого нужно задать формальное правило кодирования, согласно которому отдельные свойства реальных уязвимостей возможно оценить на шкале S_p.

Зададим **матрицу кодирования признаков уязвимостей**

$$M_{Vulners} = (p_{ij}); p_{ij} \in Z; p_{ij} \geq 0,$$

где строки матрицы $M_{Vulners}$ представляют собой отдельные свойства уязвимостей (vulner property), столбцы указывают на числовой код (code) некоторого свойства, а в ячейках p_{ij} матрицы указываются значения свойств.

Для построения такой матрицы должны быть выделены только значимые свойства, однозначно отличающие одну автоматически найденную уязвимость от другой. Понятно, что для каждого сканера информационной безопасности классификация уязвимостей может быть своей. Тем не менее большинство из них содержат такие свойства как, например, тип уязвимости, протокол, по которому она может быть эксплуатирована, канал реализации внутри этого протокола, тип уязвимого объекта, путь до объекта на сервере, сетевой запрос с вектором атаки и т.д. Все возможные значения каждого свойства кодируются неотрицательными целыми числами, где ноль выделен в качестве неопределённого значения свойства, что позволит учитывать в том числе отсутствующие, новые или пока не предусмотренные значения.

Code: / *Vulner property:*	0	1	2	3	4	. . .
Type	unknown	XSS	SQLi	LFI	. . .	
Protocol	unknown	HTTP/1.1	FTP	. . .		
Channel	unknown	get	post	cookie	url	. . .
Object type	unknown	php	js	html	. . .	
Object path	unknown	/source	/user	/upload	. . .	
. . .						

Рис. 5. Пример задания таблицы MVulners для кодирования признаков уязвимостей

Матрица $M_{Vulners}$ может быть представлена в табличном виде (см. рис. 5). Значениями свойств могут быть также нечёткие величины, и для использования в дальнейших расчётах их нужно дефазифицировать при помощи функции *Defuz*. После кодирования уязвимостей для отображения значений из множества неотрицательных целых чисел на шкалу S_p можно использовать функцию конвертирования ***Convertor*** [3, с. 67]:

$$\forall M_{Vulners}, \quad \forall j, \quad Convertor(j) = \begin{cases} \frac{1}{\pi}\operatorname{arctg}(j) + \frac{1}{2}, \; j \in Z, \; j \geq 0, \; j < \max(j), \\ 1, \; j = \max(j). \end{cases}$$

Таким образом, формальное правило кодирования свойств v_i уязвимости $v \in Vulners$ можно записать следующим образом:

$$Coding : Vulners \times M_{Vulners} \times Convertor \rightarrow S_p.$$

Предложенный способ кодирования вполне подходит для решения поставленных задач. Поиск наилучшего способа числового кодирования свойств уязвимостей не является целью данного исследования.

Анализ способов решения задачи классификации уязвимостей

Пусть $c = \{c_i\} \in Candidates$ – вектор признаков некоторой уязвимости, которую требуется отнести к I или II классам. Будем считать, что все значения параметров c_i – чёткие, закодированные функцией *Coding* и представленные на шкале $S_p = [0, 1]$. Если вектор признаков c задан нечёткими параметрами, то значения c_i должны быть дефазифицированы при помощи функции $Defuz(c_i)$. Пусть также заданы $e_k \in Eth$ – векторы признаков эталонных уязвимостей из I класса, с которыми будут сравниваться остальные векторы.

Первый способ классификации уязвимости c может заключаться в построении решающего правила *Classificator* на основе сравнения расстояния ρ между вектором c и всеми векторами e_k с пороговым значением *delta*:

$$Classificator(c, Eth): c \in Ver \Leftrightarrow \forall e_k \in Eth \quad \rho(c, e_k) \le delta.$$

В качестве расстояния ρ может использоваться любая подходящая метрика для векторного n-мерного пространства *Vulners*, например, евклидова:

$$\rho(c, e) = \sqrt{\sum_{i=1}^{n} (c_i - e_i)^2}\ .$$

Очевидные положительные стороны такого подхода – это понятный алгоритм и простота его реализации. Подобные алгоритмы успешно используются при решении множества практических задач. Однако недостатком такого подхода для решения задачи классификации уязвимостей является то, что на практике множество векторов признаков даже одного типа не будут образовывать непрерывного и ограниченного подпространства пространства *Vulners* (в смысле подпространства для R^n). Это связано с тем, что автоматические сканеры не всегда могут однозначно определить некоторые свойства уязвимостей при сканировании веб-приложений. То есть при использовании матрицы кодирования $M_{Vulners}$ даже для однотипных уязвимостей нам придется иногда ставить нули для значений некоторых свойств, либо они могут сильно отличаться от аналогичных эталонных значений. А так как метрика расстояния ρ довольно чувствительна к изменениям параметров, то при сравнении векторов признаков уязвимостей с эталонами расстояние между ними может сильно отличаться от *delta*.

Вторым способом классификации уязвимостей может стать построение цепочки решающих правил вида ЕСЛИ ..., ТО ..., в которых возможно учесть любые изменения значений параметров. Рассуждения эксперта при этом выглядят следующим образом. Если среди эталонных уязвимостей существует вектор e_k со значениями параметров α_i, и при этом вектор признаков уязвимости c имеет значения параметров c_i, то этот вектор принадлежит к I классу подтверждённых уязвимостей:

$$Classificator(c, Eth): c = \{c_i\} \in Ver \Leftrightarrow \exists e_k \in Eth,\ e_k = \{\alpha_i\}.$$

Положительные стороны такого подхода заключаются в том, что правила позволяют более гибко описать процесс принятия решения для классификатора. Однако на практике количество правил может оказаться слишком велико. Даже если совместить оба описанных способа в один: использовать в классификаторе цепочку правил, в которых выполняется сравнение значений свойств уязвимости и эталонов с некоторыми пороговыми значениями, – то всё равно реализация такого алгоритма будет затруднительна как при разработке, так и при поддержке правил в актуальном состоянии.

Недостатки указанных выше решающих правил можно попытаться устранить при помощи современного математического аппарата нейронных сетей, которые хорошо себя зарекомендовали при решении множества сложных практических задач: распознавания образов, прогнозирования, классификации и других [4].

Для решения нашей задачи необходимо определить нейронную сеть $\boldsymbol{Neuronet_{Eth}}(\boldsymbol{Config}, \boldsymbol{c})$, с конфигурацией *Config*, обученную на векторах признаках эталонных уязвимостей из *Eth*, а в качестве решающего правила использовать значения выходов нейронной сети для уязвимости-кандидата:

$$Classificator(c, Eth): Neuronet_{Eth}(Config, c) \to (s_I, s_{II});$$

$$Config = \text{const};\ c \in Candidates;\ s_I, s_{II} \in S_p.$$

Такой классификатор позволит более точно определить значения уровней принадлежности уязвимостей I или II классу, так как решающие правила фактически будут построены при обучении нейронной сети – она «запомнит» эталонные образцы, даже если значения отдельных признаков в них будут сильно отличаться. На вход нейронной сети должны подаваться значения векторов признаков уязвимостей, закодированных при помощи функции *Coding*. Построим далее нейронную сеть *Neuronet* с подходящей для наших целей конфигурацией *Config*.

Построение нейронной сети, её обучение и представление результатов

Пусть нам задано конечное число M векторов признаков уязвимостей-кандидатов, каждый из которых обладает N свойствами:

$$Candidates = \left\{ c_i^j \right\} \subset Vulners, \ i = \overline{1,N}, \ j = \overline{1,M}.$$

Также нам задано конечное число K векторов признаков эталонных уязвимостей, каждый из которых также обладает N свойствами:

$$Eth = \left\{ e_i^k \right\} \subseteq Ver, \quad i = \overline{1,N}, \quad k = \overline{1,K}.$$

Будем задавать конфигурацию нейронной сети тройкой значений:

$$Config = \left\langle inputs, \left\{ layer^l \right\}, outputs \right\rangle, l = \overline{1,L},$$

где *inputs* – количество входных параметров; $\{layer^l\}$ – множество неотрицательных целых чисел, указывающих на количество нейронов в скрытом слое номер l; L – число слоёв; *outputs* – количество выходных параметров.

Будем считать, что используется полносвязная многослойная сеть персептронов: выход каждого нейрона в каждом слое связан со входами всех нейронов следующего слоя.

Для решения нашей задачи использовалась конфигурация (см. рис. 6):

$$Config = \left\langle N, \left\{ N, \left[\frac{N}{2} \right] \right\}, 2 \right\rangle,$$

где $inputs = N$ – по количеству свойств уязвимостей; $L = 2$ – установленное эмпирическим путем достаточное значение слоёв нейронной сети для классификации уязвимостей; $\{ layer^{1,2} \} = \{N, [N/2] \}$ – установленное эмпирическим путем число нейронов в первом скрытом слое должно совпадать с количеством входных параметров, а во втором – быть целой частью от половины этого количества; $outputs = 2$ – по количеству классов уязвимостей, так как на выходе мы получаем вектор (s_I, s_{II}) с оценками принадлежности классам *Ver* и *NVer*.

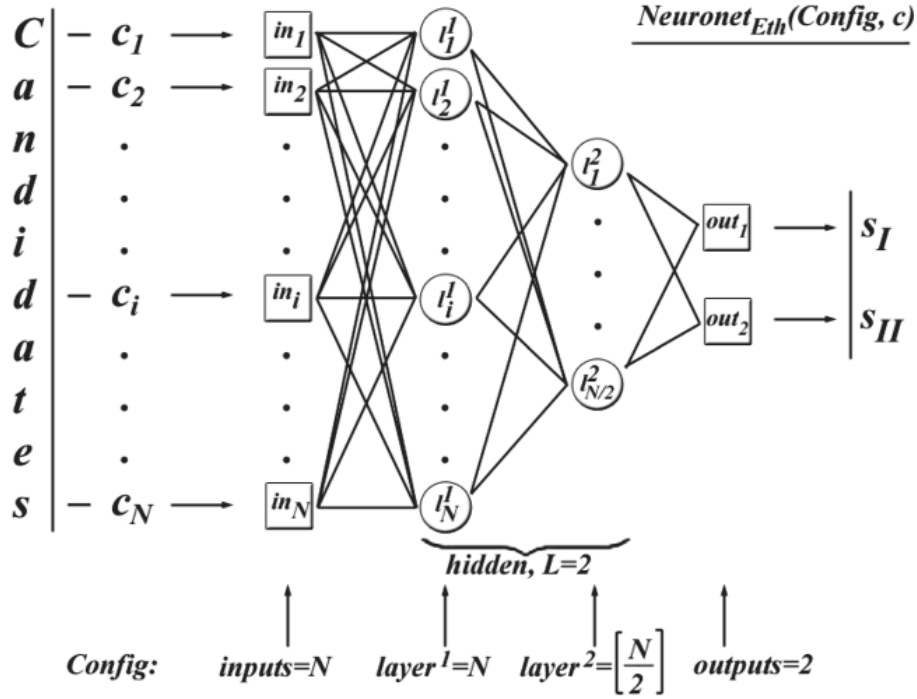

Рис. 6. Структура нейронной сети NeuronetEth(Config, c) для задачи классификации

Известный алгоритм обучения нейронной сети с учителем заключается в итеративном подборе весовых коэффициентов w_i для каждого нейрона сети таким образом, чтобы значение вектора (s_I, s_{II}) на её выходе совпадало с требуемым значением для

каждого входного вектора признаков из *Eth* [4, п.п. 2.1]. Подбор весов повторяется до тех пор, пока не устраняются противоречия для всех эталонов, либо ошибка такого обучения становится минимальной.

Вектор (s_I, s_{II}) со значениями параметров на шкале S_p может интерпретироваться следующим образом:

1. Значения параметров указывают на степень уверенности от 0 до 1 в принадлежности вектора признаков уязвимости каждому классу.

2. Значения параметров, будучи умноженными на 100%, указывают на вероятность принадлежности вектора признаков уязвимости каждому классу от 0 до 100%.

3. Значения параметров, фазифицированные при помощи функции $Fuzzy(x, S_f)$, указывают на лингвистическую оценку уровня принадлежности вектора признаков уязвимости каждому из классов на шкале $S_f = \{Min, Low, Med, High, Max\}$.

На практике может использоваться любая из этих интерпретаций, удобная эксперту-аналитику.

Программная реализация классификатора

Для практического использования нейронных сетей при решении задач нечёткой классификации в случае различного числа классов и структуры сетей были разработаны программные модули **FuzzyClassificator**, распространяемые под лицензией GNU GPL v3.

Скачать актуальную версию FuzzyClassificator можно по ссылке на GitHub: **https://github.com/Tim55667757/FuzzyClassificator/tree/master**.

Для удобства использования модулей в системах автоматизации работа с программой осуществляется через интерфейс командной строки. В разделе описания программы на GitHub приведена вся информация по командам интерфейса, работе модулей и заданию входных данных. FuzzyClassificator для своей работы требует **Pyzo** – бесплатный и открытый инструмент разработки, основанный на Python 3.3.2 и включающий в себя множество подпрограмм для реализации научных вычислений, в частности **PyBrain library** – подпрограммы для работы с нейронными сетями.

Основные программные модули, реализующие предложенные в статье подходы и математический аппарат, следующие:

1. FuzzyClassificator – реализует пользовательский интерфейс командной строки, получает и обрабатывает входные данные, устанавливает режимы обучения и классификации, предоставляет результаты.

2. PyBrainLearning – определяет методы для работы с нечёткими нейронными сетями, объединяя возможности библиотеки PyBrain и авторской библиотеки FuzzyRoutines.

3. FuzzyRoutines – содержит подпрограммы для работы с нечёткими множествами и нечёткими шкалами.

Процесс работы программы представлен функциональной IDEF0-моделью на рис. 7–9.

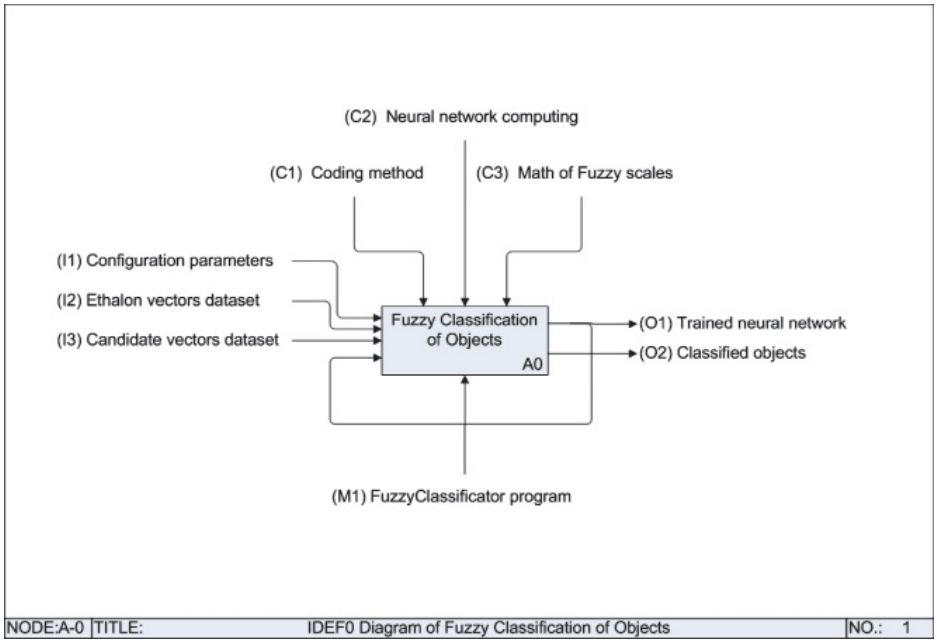

Рис. 7. Верхний A–0 уровень функциональной IDEF0-модели процесса работы программы FuzzyClassificator

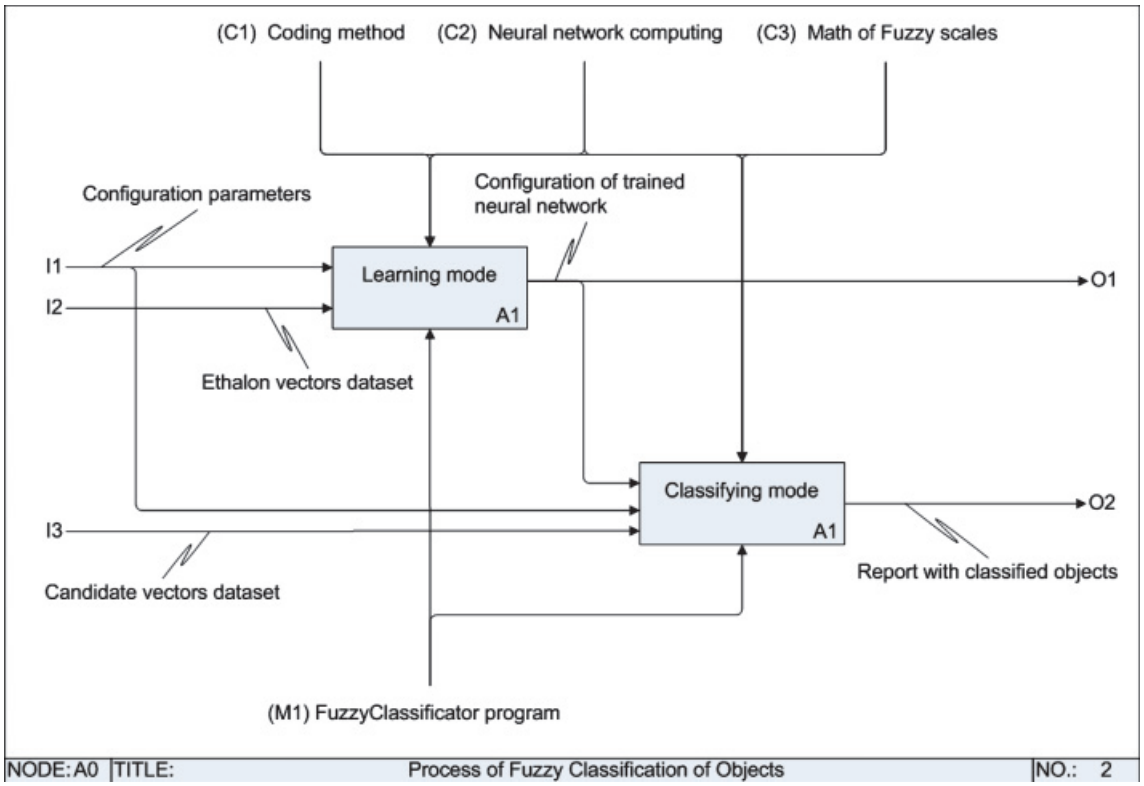

Рис. 8. Уровень A0 IDEF0-модели.
Основные этапы работы программы FuzzyClassificator

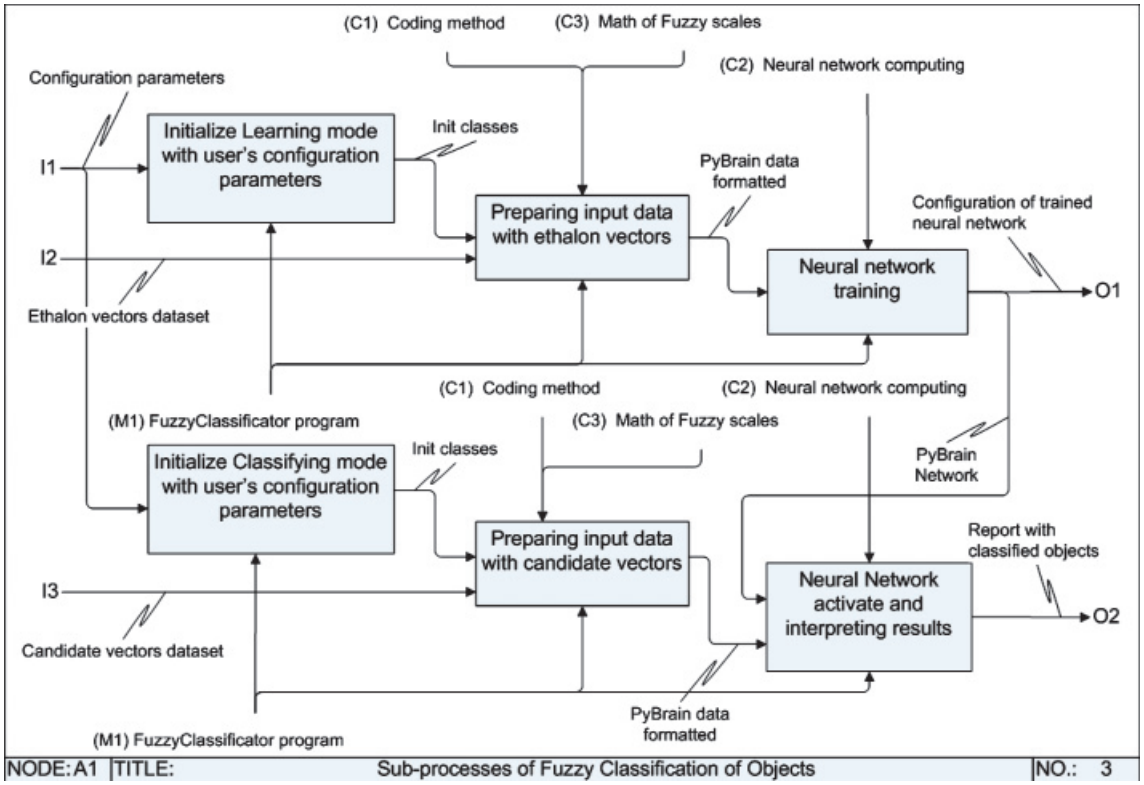

Рис. 9. Уровень A1 IDEF0-модели. Разбиение этапов работы программы FuzzyClassificator на подпроцессы

1. Этап обучения (Learning mode) состоит из следующих шагов:

1.1. Инициализация объектов программы значениями от пользователя.

1.2. Обработка входных данных и подготовка нейросети к обучению:

– обработка файла с данными о векторах признаков эталонов;

– подготовка данных для обучения в формате PyBrain;

– инициализация параметров новой нейронной сети PyBrain или её загрузка из указанного файла.

1.3. Обучение нейронной сети на заданных эталонах:

– инициализация модуля PyBrain-тренера;

– обучение сети при помощи тренера и сохранение её конфигурации в файл формата PyBrain.

2. Этап классификации (Classifying mode) состоит из следующих шагов:

2.1. Инициализация объектов программы значениями от пользователя.

2.2. Обработка входных данных и подготовка нейросети к их анализу:

– обработка файла с данными о векторах признаков кандидатов;

– загрузка конфигурации обученной нейронной сети PyBrain из указанного файла.

2.3. Анализ нейронной сетью векторов признаков кандидатов:

– активация нейронной сети и вычисление уровней принадлежности векторов к различным классам;

– интерпретация полученных результатов на нечётких шкалах и формирование файла отчёта.

Входные данные с векторами признаков эталонов и кандидатов задаются в виде обычных текстовых файлов с табуляцией в качестве разделителя значений. Например, чтобы задать данные для обучения, можно подготовить файл ethalons.dat, содержащий строку заголовка и далее строки со значениями эталонных векторов признаков и их принадлежности тому или иному классу. Значения могут быть заданы как на чёткой, так и на нечёткой шкалах:

input1	**input2**	**input3**	**1st_class_output**	**2nd_class_output**
0.1	0.2	Min	**Min**	**Max**
0.2	0.3	Low	**Min**	**Max**
0.3	0.4	Med	**Min**	**Max**
0.4	0.5	Med	**Max**	**Min**
0.5	0.6	High	**Max**	**Min**
0.6	0.7	Max	**Max**	**Min**

А в качестве данных для анализа может быть подготовлен файл candidates.dat, также содержащий строку заголовка и строки со значениями векторов признаков кандидатов:

input1	**input2**	**input3**
0.12	0.32	Min
0.32	0.35	Low
0.54	0.57	Med
0.65	0.68	High
0.76	0.79	Max

По итогам работы программы создаётся файл с отчётом, содержащим информацию о конфигурации нейронной сети и результаты классификации для каждого вектора признаков из множества кандидатов.

После обучения нейронной сети на указанных выше примерах с параметрами, заданными командной строкой

```
python FuzzyClassificator.py --ethalons ethalons.dat --learn
config=3,3,2,2 epochs=1000 rate=0.1 momentum=0.05
```

и затем, в режиме классификации с параметрами командной строки

```
python FuzzyClassificator.py --candidates candidates.dat --network
network.xml --report report.txt --classify config=3,3,2,2
```

на выходе был получен следующий репорт-файл:

```
Neuronet: C:\work\projects\FuzzyClassificator\network.xml

FuzzyScale = {Min, Low, Med, High, Max}
    Min = <Hyperbolic(x, {'a': 8, 'c': 0, 'b': 20}), [0.0, 0.23]>
    Low = <Bell(x, {'a': 0.17, 'c': 0.34, 'b': 0.23}), [0.17, 0.4]>
    Med = <Bell(x, {'a': 0.34, 'c': 0.6, 'b': 0.4}), [0.34, 0.66]>
    High = <Bell(x, {'a': 0.6, 'c': 0.77, 'b': 0.66}), [0.6, 0.83]>
    Max = <Parabolic(x, {'a': 0.77, 'b': 0.95}), [0.77, 1.0]>

Classification results for candidates vectors:

    Input: ['0.12', '0.32', 'Min']    Output: ['Min', 'Max']
    Input: ['0.32', '0.35', 'Low']    Output: ['Low', 'High']
    Input: ['0.54', '0.57', 'Med']    Output: ['Max', 'Min']
    Input: ['0.65', '0.68', 'High']   Output: ['Max', 'Min']
    Input: ['0.76', '0.79', 'Max']    Output: ['Max', 'Min']
```

Если проанализировать данные из файла candidates.dat, то можно с высокой степенью уверенности утверждать, что человек-эксперт, опираясь только на данные из файла ethalons.dat, выдал бы аналогичные результаты классификации.

Заключение

Итак, благодаря проведённому исследованию удалось совместить математические аппараты теорий нечётких систем и нейронных сетей для решения практической задачи классификации уязвимостей. В данной статье предлагается универсальный способ представления входных данных в виде вектора признаков уязвимостей $v = \{v_i\}$. Для его кодирования вводится матрица $M_{Vulners}$. Значения свойств уязвимостей оцениваются как на чёткой $S_p = [0, 1]$, так и на универсальной нечёткой шкале $S_f = \{Min, Low, Med, High, Max\}$. Вводятся функции связи между шкалами: $Fuzzy(x, S_f)$ и $Defuz(A_i)$. Классификатор уязвимостей определяется через построение нейронной сети $Neuronet_{Eth}(Config, c)$ с конфигурацией $Config = \langle inputs, \{layer^j\}, outputs\rangle$ и непустым множеством эталонов Eth. Для практического использования классификатора разработаны универсальные программные модули FuzzyClassificator, позволяющие выполнять нечёткую классификацию произвольных объектов с неограниченным числом свойств, классов и произвольной конфигурацией многослойной нейронной сети на базе персептронов.

Основные выводы

1. Математические методы разделения на классы на базе нейронных сетей применимы и в случае классификации уязвимостей.

2. Для получения адекватных результатов необходимо корректно построить матрицу кодирования и подобрать наилучшие свойства для моделирования уязвимостей.

3. Для задачи классификации уязвимостей рекомендуется использовать нейронную сеть персептронов с двумя скрытыми слоями и в конфигурации, зависящей от числа входных параметров: в первом число нейронов равно числу входных параметров, а во втором – в два раза меньше.

4. Преимуществом предложенных подходов является использование универсальных нечётких шкал лингвистических переменных, которые применимы как для оценки значений векторов признаков, так и для интерпретации итоговых уровней принадлежности классам.

5. Предложенный метод нечёткой классификации и реализующие его программные модули FuzzyClassificator являются универсальными, легко адаптируются и настраиваются под конкретные объекты классификации.

Список литературы

1. Адлер Ю.П., Маркова Е.В., Грановский Ю.В. Планирование эксперимента при поиске оптимальных условий. – М.: Наука, 1976. – 280 с.

2. Алтунин А.Е., Семухин М.В. Модели и алгоритмы принятия решений в нечётких условиях: монография. – Тюмень: Изд-во Тюменского гос. ун-та, 2000. – 352 с.

3. Гильмуллин Т.М. Модели и комплекс программ процесса управления рисками информационной безопасности: дис. ... канд. технич. наук. – Казанский гос. технический ун-т. – Казань, 2010. – 225 с. URL: https://drive.google.com/file/d/0B-1rf8K04ZS5WjJyWFFBeW91YTQ/edit?usp=sharing.

4. Гильмуллин Т.М. Применение нейросетей для решения классических задач линейного и нелинейного разделения элементов множества на классы // Лабораторные работы для дисциплины «Нейрокомпьютерные системы. – 2008», КГТУ им. А.Н. Туполева (КАИ), 2013. URL: http://math-n-algo.blogspot.ru/2013/04/blog-post.html.

5. Гильмуллин Т.М. Тестирование сканеров безопасности веб-приложений: подходы и критерии // habrahabr.ru, 2013. URL: http://habrahabr.ru/company/pt/blog/187636.

6. Shay C. WAVSEP Web Application Scanner Benchmark 2014 // WAVSEP 2013/2014 Score Chart: The Web Application Vulnerability Scanners Benchmark. An Accuracy, Coverage, Versatility, Adaptability, Feature and Price Comparison of 63 Black Box Web Application Vulnerability Scanners and SAAS Services, 2014. URL: http://sectooladdict.blogspot.ru/2014/02/wavsep-web-application-scanner.html.

References

1. Adler Ju.P., Markova E.V., Granovskij Ju.V. Planirovanie jeksperimenta pri poiske optimalnyh uslovij. M.: Nauka, 1976. 280 p.

2. Altunin A.E., Semuhin M.V. Modeli i algoritmy prinjatija reshenij v nechjotkih uslovijah: Monografija. – Tjumen: Izd-vo Tjumenskogo gos. un-ta, 2000. 352 p.

3. Gilmullin T.M. Modeli i kompleks programm processa upravlenija riskami informacionnoj bezopasnosti: dis. ... kand. tehnich. nauk. – Kazanskij gos. tehnicheskij un-t. – Kazan, 2010. 225 p. URL: https://drive.google.com/file/d/0B-1rf8K04ZS5WjJyWFFBeW91YTQ/edit?usp=sharing.

4. Gilmullin T.M. Primenenie nejrosetej dlja reshenija klassicheskih zadach linejnogo i nelinejnogo razdelenija jelementov mnozhestva na klassy // Laboratornye raboty dlja discipliny «Nejrokompjuternye sistemy. – 2008», KGTU im. A.N. Tupoleva (KAI), 2013. URL: http://math-n-algo.blogspot.ru/2013/04/blog-post.html.

5. Gilmullin T.M. Testirovanie skanerov bezopasnosti web-prilozhenij: podhody i kriterii // habrahabr.ru, 2013. URL: http://habrahabr.ru/company/pt/blog/187636.

6. Shay C. WAVSEP Web Application Scanner Benchmark 2014 // WAVSEP 2013/2014 Score Chart: The Web Application Vulnerability Scanners Benchmark. An Accuracy, Coverage, Versatility, Adaptability, Feature and Price Comparison of 63 Black Box Web Application Vulnerability Scanners and SAAS Services, 2014. URL: http://sectooladdict.blogspot.ru/2014/02/wavsep-web-application-scanner.html.

Рецензенты:

Райхлин В.А., д.ф.-м.н., профессор кафедры компьютерных систем Казанского национального исследовательского технического университета им. А.Н. Туполева – КАИ, г. Казань;

Шарнин Л.М., д.т.н., профессор, зав. кафедрой АСОИУ Казанского национального исследовательского технического университета им. А.Н. Туполева – КАИ, г. Казань.

Работа поступила в редакцию 06.10.2014.