УДК 004.4 doi: 10.15827/0236-235X.149.089-099 2025. Т. 38. № 1. С. 89–99

Авторская метрика оценки близости программ: приложение для поиска уязвимостей с помощью генетической деэволюции

М.В. Буйневич [1], К.Е. Израилов [2]✉

[1] Санкт-Петербургский университет Государственной противопожарной службы МЧС России, г. Санкт-Петербург, 196105, Россия

[2] Санкт-Петербургский Федеральный исследовательский центр РАН, г. Санкт-Петербург, 199178, Россия

Ссылка для цитирования

Буйневич М.В., Израилов К.Е. Авторская метрика оценки близости программ: приложение для поиска уязвимостей с помощью генетической деэволюции // Программные продукты и системы. 2025. Т. 38. № 1. С. 89–99. doi: 10.15827/0236-235X.149.089-099

Информация о статье

Группа специальностей ВАК: 2.3.6

Поступила в редакцию: 10.06.2024 После доработки: 25.06.2024 Принята к публикации: 28.06.2024

Аннотация. Актуальность темы статьи обусловлена наличием в сфере информационной безопасности задач, требующих сравнения программ в их различных представлениях, таких, как текстовый ассемблерный код (например, для поиска уязвимостей или подтверждения авторства). В работе представлена метрика близости двух текстов в виде списка строк из символов, являющаяся развитием ее предыдущей версии. Основным результатом текущего исследования (как части главного, направленного на генетическую деэволюцию программ) являются сама метрика, а также ее характеристики и особенности, выявленные с помощью проведенных экспериментов. Метрика представлена в аналитическом виде, программно реализована на языке Python, принимает на вход два списка символьных строк для сравнения и коэффициенты учета позиции ее элементов от начала списка и последовательности символов. Результатом ее вычисления является числовое значение в диапазоне от 0 до 1. Новизна метрики заключается в достаточно точной и чувствительной оценке близости двух текстов независимо от форматов представления данных; текущая версия метрики отличается от предыдущей учетом указанных коэффициентов. Теоретическая значимость заключается в развитии способов сравнения произвольных текстов, представляющих собой список символьных строк, содержащих информацию, последовательно излагаемую согласно определенной логике (что требует учета позиции). Помимо общего назначения сравнительных инструментов такого рода, практическая значимость метрики состоит в возможности определения близости двух программ, имеющих бинарное представление машинного кода, предварительно преобразованное в текстовое представление ассемблерного кода.

Ключевые слова: информационная безопасность, метрика близости, программное обеспечение, поиск уязвимостей, базовый принцип, генетическая деэволюция

Благодарности. Работа выполнена при частичной финансовой поддержке бюджетной темы FFZF-2025-0016

Введение. Небезопасность ПО является существенной угрозой для сферы информационных технологий. Для ее минимизации (а в идеале ликвидации) требуется решение большого количества частных, но важных научно-практических задач. Одной из таких задач является необходимость сравнения двух программ по некоторой шкале – нахождение их близости (или степени подобия). Для этого требуется создание соответствующей метрики близости (далее – метрика) как некоторого расстояния в условном пространстве программного кода [1]. Актуальность задачи определяется способами применения и областями приложения ее решений.

Во-первых, близость машинного кода используемой программы и содержащегося в базе вредоносного ПО позволяет оценивать небезопасность первой (например, с применением частотного распределения байтов или нечетких хэшей [2]). Однако у данного способа обнаружения вредоносного ПО имеется ряд проблем, связанных с существенной зависимостью точной последовательности байтов инструкций от множества факторов, включая даже ключи компиляции исходного кода программ. Применение же злоумышленником техник обфускации [3], предназначенных для существенного изменения кода программы без потери ее функциональности, вообще сделает бесполезной попытку поиска машинного кода исследуемой программы по базе вредоносного ПО.

Во-вторых, применение степени близости программ заключается в поиске дубликатов кода, содержащих и уязвимости [4]. Для этого, в частности, может применяться анализ последовательностей инструкций машинного кода с построением графа потока управления [5].

В-третьих, расширение способа поиска дубликатов путем учета более сложных синтаксических и семантических подобий даст возможность определения авторства программы как для всего исходного кода, так и для его частей [6].

Предполагаемым применением решения задачи является ее использование в исследованиях [7, 8], направленных на получение более ранних и высокоуровневых представлений программы (например, исходного кода, алгоритмов и архитектуры) по ее низкоуровневым представлениям (например, по машинному коду). Для этого предлагается применять так называемый генетический реверс-инжиниринг, состоящий из типовых генетических деэволюций ближайших представлений программы [7, 8]. Данный подход является развитием генетических алгоритмов, представляющих собой один из способов решения оптимизационной задачи в виде максимизации (или минимизации) некоторой функции путем итеративного подбора ее аргументов, которыми в данном случае определяют некоторый экземпляр исходного кода. Одно из центральных вычислений в генетическом алгоритме – функция приспособленности, которая заключается во взаимной близости двух программ в одном представлении (например, двух экземпляров машинного кода) – то есть метрику. Она и является искомым решением задачи и предметом настоящего научного исследования.

Обзор исследований

Приведем обзор научных публикаций, в которых предлагаются различные способы сравнения и оценки близости массивных символьных сущностей (данных), не ограничиваясь только выполняемыми программами, их исходным и машинным кодами.

Классической мерой сходства двух множеств считается индекс Жаккара, вычисляемый как отношение мощностей (то есть размеров) пересечения множеств к их объединению [9]; впрочем, для формулы индекса есть и другие вариации. Так, для идентичных множеств индекс равен 1, а для содержащих абсолютно разные элементы – 0.

Другим подходом к оценке близости элементов множеств является их кластеризация различными алгоритмами (например, k-средних, семейство FOREL и др.). Соответственно, элементы одного кластера считаются близкими по некоторым признакам [10].

Оценивание близости программ к случайным последовательностям байт позволяет идентифицировать данные как зашифрованные. Для этого, помимо вычисления энтропии данных, предлагается применять вейвлет-преобразования.

Близость машинного кода программ для процессорных платформ x86, ARM и MIPS в исследовании [11] оценивается через семантику машинных инструкций, обрабатываемых в процессе выполнения или эмуляции.

В работе [12] оценивается близость текстов, представленных в виде структурированных документов, с применением смыслового анализатора. Для этого, в частности, выделяются основные понятия в тексте и их онтологические связи, задаются параметры анализа (сила и вес связи), проверяется совпадение терминологий сравниваемых документов, строится семантическая сеть документов и оценивается их соответствующая (семантическая) близость.

Уже классикой считается оценка близости документов на основании их тематической сепарации с применением кластеризации.

Описание различных представлений матриц близости на основе пространственной, теоретико-множественной и графовой моделей, а также их объединения дано в работе [13]. В качестве основного применения моделей указывается анализ социологических данных.

Исследование [14] посвящено поиску дубликатов графов. Для этого предлагается оригинальный алгоритм, основанный на индексации графов, а также на различных мерах их схожести.

Как показал краткий обзор опубликованных результатов научных исследований, все они или являются слишком общими и прогнозируемо имеющими низкую чувствительность для сравнения программ (например, применение индекса Жаккара или тематическая сепарация), или же, наоборот, учитывают частные аспекты данных и обладают слабой инвариантностью ко всему множеству процессоров выполнения машинного кода, языков программирования исходного кода и т.п. (например, использование семантики машинных инструкций). Предлагается авторская метрика, гипотетически применимая практически для любых текстовых и бинарных представлений двух программ и при этом обладающая высокой чувствительностью к их различиям.

Описание метрики

Требования. Основное предназначение метрики, помимо применения для решения различных частных задач информационной безопасности, обусловлено необходимостью создания авторского способа генетической деэволюции представлений программ. При этом наиболее классической деэволюцией считается преобразование программы из машинного кода (из бинарного представления) в исходный (в текстовое представление). Последнее может быть проанализировано экспертом по информационной безопасности программы на предмет наличия уязвимостей. Процесс получения текстового представления инструкций машинного кода, то есть ассемблерного кода программы, является хорошо отработанным и реализуемым с помощью специальных утилит – дизассемблеров [15]. Таким образом, метрика может оперировать именно текстами ассемблерного кода.

Принцип вычисления метрики для текстов (то есть списков из последовательностей символов) X и Y заключается в оценке близости между всеми строками из X и Y. Затем выбирается максимальное значение близости между каждой строкой из X и всеми строками из Y, которое корректируется с учетом расстояния между позициями этих строк в каждом тексте. Тем самым учитывается не только близость строк, но и их отдаление от одинаковых позиций. После этого производится средняя оценка близости всех строк из текстов. Аналогичным образом принцип вычисления метрики применяется и для двух строк только на основании совпадения и нахождения на одинаковых позициях пар символов. При этом близость дополнительно корректируется с учетом дальности расположения символов и строк от начала их последовательностей и текстов. Влияние дальности определяется соответствующими коэффициентами, учитываемыми метрикой. Отсутствие сравниваемой строки в одном из списков, как и отсутствие сравниваемого символа в одной из последовательностей, может трактоваться как их нахождение на большом расстоянии, равном максимальному размеру одного из списков и максимальной длине одной из строк соответственно. Для удобства интерпретации значения метрики целесообразно отнормировать его к диапазону [0,1], например, путем суммирования всех близостей пар строк из X и $У$ и их деления на количество строк в списке. Естественно, близость строк должна нормироваться к такому же диапазону. В этом случае при полном несовпадении строк из X и Y вплоть до отсутствия общих символов их попарные оценки близостей будут равны 0, что при суммировании даст значение метрики, также равное 0. Для обратной ситуации, когда все строки в X и Y полностью совпадают и находятся на тождественных позициях, попарные близости дадут значение 1, а их сумма будет равна размеру списков, на который и необходимо будет поделить итоговый результат для его нормировки. Приведенный принцип может считаться базовым, поскольку он отражает суть вычисления метрики и может использоваться для ее качественного сравнения с аналогами.

Аналитическая модель. Для формализации метрики предлагается аналитическая модель ее вычисления, соответствующая описанному принципу.

Основная формула близости двух строк:

$$\begin{cases} Metric^{Strings}_{str1,\ str2} = \dfrac{1}{length^{Strings}} \times \\ \times \sum\limits_{i1\in[1\ldots length(str1)]} \max\left(\bigcup\limits_{i2\in[1\ldots length(str2)]} metric^{Symbols}_{i1,\ i2} \right), \\ length(str1) \geq length(str2) \end{cases} \quad (1)$$

где $Metric^{Strings}_{str1,\ str2}$ – близость двух строк $str1$ и $str2$, притом что 1-я строка выбирается более длинной или такой же, как 2-я строка; $length^{Strings}$ – максимальная длина (в символах) из двух строк; $i1$ и $i2$ – индекс символа каждой из строк в диапазоне от 1 до последнего, равного длине строки; $length(\ldots)$ – операция получения длины (в данном случае – строки из символов) и таким образом $length^{Strings} = length(str1)$; $\max(\ldots)$ – операция нахождения максимального элемента в множестве; $metric^{Symbols}_{i1,i2}$ – частная метрика близости двух символов с индексами $i1$ и $i2$.

Таким образом, метрика близости двух строк вычисляется как сумма частных метрик максимальных близостей символа 1-й строки и всех символов 2-й строки (такое усложнение расчета обусловлено тем, что некоторый символ в 1-й строке может присутствовать в нескольких позициях 2-й строки и необходимо выбрать наиболее близкий). Указанная частная метрика близости двух символов определяется следующим образом:

$$metric^{Symbols}_{i1,i2} = \left[\begin{array}{l} 1 - \dfrac{diff^{SymbolCorr}_{i1,i2}}{length}, \\ \text{если}\ str1[i1] = str2[i2], \\ 0,\ \text{если}\ str1[i1] \neq str2[i2], \end{array} \right. \quad (2)$$

где $diff^{SymbolCorr}_{i1,i2}$ – скорректированное (с учетом позиции или индекса относительно начала строк) расстояние между позициями некоторого символа в разных строках; $str[i]$ – символ в строке str на i-й позиции.

Таким образом, если символы в $i1$-й позиции 1-й строки и $i2$-й позиции 2-й строки различны, то метрика будет минимальной – равной 0; а, например, если в одинаковых позициях (то есть $i1 = i2$) находится один и тот же символ, то расстояние между ними равно 0 ($diff_{i1,i2}^{SymbolCorr} = 0$) и, следовательно, метрика будет максимальной – равной 1. В других случаях указанное скорректированное расстояние определяется по следующей формуле:

$$diff_{i1,i2}^{SymbolCorr} = diff_{i1,i2}^{Symbol} \times \left(1 + p^{SymbolDist}\left(1 - \frac{diff_{i1,i2}^{Symbol}}{length}\right)\right), \quad (3)$$

где $diff_{i1,i2}^{Symbol} = |\, i1 - i2 \,|$ – расстояние (нескорректированное по позиции) между индексами символа в разных строках; $p^{SymbolDist}$ – вычисляемый параметр корректировки расстояния между позициями символом, определяемым в диапазоне $[0\ \ldots 1]$; больший параметр соответствует большему влиянию расстояния, а 0 – отсутствию такового. Значит, при наименьшей корректировке ($p^{SymbolDist} = 0$) эти расстояния будут идентичны – $diff_{i1,i2}^{SymbolCorr} = diff_{i1,i2}^{Symbol}$. Параметр корректировки определяется следующим образом:

$$p^{SymbolDist} = \begin{cases} \dfrac{\left(avg_{i1,i2} - 1\right)k^{String}}{length - 1}, \\ если\ length \geq 2, \\ 0,\ если\ length = 1, \end{cases} \quad (4)$$

где $avg_{i1,i2} = \dfrac{i1 + i2}{2}$ – средняя позиция символа, определяемая по позициям в двух строках ($i1$ и $i2$); k^{String} – задаваемый коэффициент дополнительной корректировки близости согласно расстоянию символов от начала строк, определяемый в диапазоне $[0\ldots 1]$; больший параметр соответствует большему влиянию расстояния, а 0 – отсутствию такового.

В случае строк единичной длины ($length = 1$) согласно формуле (4) было бы деление на 0, и во избежание такой ситуации добавлено дополнительное условие, определяющее для таких строк отсутствие влияния расстояния на метрику ($p^{SymbolDist} = 0$). Если дополнительная корректировка близости отсутствует ($k^{String} = 0$), корректировка расстояния между позициями символов также будет отсутствовать ($p^{SymbolDist} = 0$).

Аналогична формула близости двух текстов (вычисляемая через близость их символов):

$$\begin{cases} Metric_{txt1,txt2}^{Texts} = \dfrac{1}{length^{Texts}} \times \\ \times \sum\limits_{i1 \in [1\ldots length(txt1)]} \max\left(\bigcup\limits_{i2 \in [1\ldots length(txt2)]} metric_{i1,i2}^{StringsCorr} \right), \\ length(txt1) \geq length(txt2) \end{cases} \quad (5)$$

где $Metric_{txt1,txt2}^{Texts}$ – близость двух текстов $txt1$ и $txt2$ (как и для $Metric_{str1,\ str2}^{Strings}$, размер первого объекта сравнения выбирается больше второго, но уже в виде количества строк); $length = (\ldots)$ – указанная ранее операция получения длины (в данном случае – текста из строк), таким образом, $length^{Texts} = length(txt1)$; $metric_{i1,i2}^{StringsCorr}$ – частная метрика близости $i1$-й строки из 1-го текста и $i2$-й строки из 2-го текста с учетом корректировок, которая определяется следующим образом:

$$metric_{i1,i2}^{StringsCorr} = \begin{cases} Metric_{str1,str2}^{Strings} \times \left(1 - \dfrac{diff_{i1,i2}^{StringCorr}}{length^{Texts}}\right), \\ если\ Metric_{str1,str2}^{Strings} = \max\left(metrics_{i1}^{String}\right) \\ 0,\ если\ Metric_{str1,str2}^{Strings} \neq \max\left(metrics_{i1}^{String}\right) \end{cases}, \quad (6)$$

где $Metric_{str1,str2}^{Strings}$ – метрика близости двух строк, описанная ранее; $diff_{i1,i2}^{StringCorr}$ – скорректированное расстояние между позициями некоторой строки в разных текстах, вычисляемое полностью аналогично $diff_{i1,i2}^{SymbolCorr}$, но с использованием вводимого коэффициента дополнительной корректировки близости согласно расстоянию строк от начала текста k^{Text}, подставляемого в формулу (6) вместо k^{String}; $metrics_{i1}^{String}$ – множество метрик близости $i1$-й строки из 1-го текста и всех строк из 2-го текста:

$$\begin{cases} metrics_{i1}^{String} = \bigcup\limits_{i2 \in [length(txt1)]} Metric_{str1,str2}^{Strings} \\ str1 = txt[i1] \\ str2 = txt[i2] \end{cases}, \quad (7)$$

где $str1$ и $str2$ – строки в текстах $txt1$ и $txt2$, расположенные на позициях $i1$ и $i2$.

Согласно базовому принципу, формулы для вычисления метрики близости текстов практически полностью идентичны формулам метрики близости строк (1)–(7) с тем дополнением, что первые в своем расчете используют вторые через формулу (1) как дополнительный параметр, способный ослаблять основную метрику.

Алгоритм вычисления, реализован на языке Python 3.11. Наиболее сложной для понимания является формула (3) для скорректированного расстояния символов (и аналогичная ей для строк) в зависимости от их начала, что можно пояснить на следующем примере. Предположим, что необходимо оценить близость символа «*x*», присутствующего в единственном экземпляре в двух строках длиной в 20 символов; при этом символ «*x*» в первой строке расположен на 10-й позиции. Тогда зависимость значения $diff_{i1,i2}^{SymbolCorr}$ при различных корректировочных коэффициентах (k^{String}) и позициях второго символа «*x*» будет такой, как представлено в таблице 1. В нее также добавлена тепловая маркировка ячеек таблицы – зеленый цвет соответствует меньшим значениям расстояния, а красный – бóльшим.

Согласно тепловой карте и значениям $diff_{i1,i2}^{SymbolCorr}$ в ячейках таблицы 1, при нахождении 2-го символа «*x*» в той же позиции, что и первый (то есть равной 10), расстояние при любых корректировочных коэффициентах равно 0. По мере увеличения расстояния между этими символами в двух строках $diff_{i1,i2}^{SymbolCorr}$ постепенно увеличивается (максимум достигается в позициях 0 и 20), при этом при более высоких k^{String} (максимум достигается при значении 1.0) данное увеличение будет более существенным. Выявленная закономерность доказывает хорошую чувствительность метрики как к расстоянию между одинаковыми символами/строками, так и к их отступу от начала последовательности/списка.

Эксперимент

Для демонстрации работоспособности и применимости предлагаемой метрики был проведен соответствующий эксперимент. Ее первая версия (далее – метрика_1), не учитывающая удаленность символов и строк от начала последовательностей и списков, уже была базово протестирована. В данном исследовании основные тестируемые аспекты были расширены для оценивания именно ее новой версии (далее – метрика_2). Ранее значение метрики_1 сопоставлялось с индексом Жаккарда и показало ее преимущество над классическим инструментом сравнения множеств. Сравним метрику_2 с метрикой_1, что позволит увидеть ее новизну, особенности и преимущества.

Все тесты были поделены на четыре группы. Поскольку алгоритмы вычисления близости строк алгоритмически подобны таким же алгоритмам для близости текстов, тесты будут приведены только для первых – они полностью соответствуют результатам тестирования вторых. Вводимый коэффициент дополнительной корректировки близости строк (k^{String}) соответствовал незначительному влиянию позиции символов относительно начала строки (равнялся 0.1), а значение коэффициента дополнительной корректировки текстов (k^{Text}) не учитывалось (равнялось 0.0).

Таблица 1

Зависимость $diff_{i1,i2}^{SymbolCorr}$ от k^{String} и позиции символа «*x*» во второй строке

Table 1

Dependence of $diff_{i1,i2}^{SymbolCorr}$ on k^{String} and the character "*x*" position in the 2nd line

Коэффициент корректировки (k^{String})	Позиция символа «*x*» во второй строке ($diff_{i1,i2}^{SymbolCorr}$)																				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	10.0	9.0	8.0	7.0	6.0	5.0	4.0	3.0	2.0	1.0	0.0	1.0	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0
0.1	10.1	9.1	8.2	7.2	6.2	5.2	4.1	3.1	2.1	1.1	0.0	1.1	2.1	3.2	4.2	5.3	6.3	7.3	8.4	9.4	10.4
0.2	10.3	9.3	8.3	7.3	6.3	5.3	4.3	3.2	2.2	1.1	0.0	1.1	2.2	3.3	4.4	5.5	6.6	7.7	8.7	9.8	10.8
0.3	10.4	9.4	8.5	7.5	6.5	5.4	4.4	3.3	2.3	1.1	0.0	1.2	2.3	3.5	4.6	5.7	6.9	8.0	9.1	10.1	11.2
0.4	10.5	9.6	8.6	7.6	6.6	5.6	4.5	3.5	2.3	1.2	0.0	1.2	2.4	3.6	4.8	6.0	7.2	8.3	9.4	10.5	11.6
0.5	10.7	9.7	8.8	7.8	6.8	5.7	4.7	3.6	2.4	1.2	0.0	1.3	2.5	3.8	5.0	6.2	7.4	8.6	9.8	10.9	12.0
0.6	10.8	9.9	8.9	7.9	6.9	5.9	4.8	3.7	2.5	1.3	0.0	1.3	2.6	3.9	5.2	6.5	7.7	8.9	10.1	11.3	12.4
0.7	10.9	10.0	9.1	8.1	7.1	6.0	4.9	3.8	2.6	1.3	0.0	1.4	2.7	4.1	5.4	6.7	8.0	9.3	10.5	11.6	12.8
0.8	11.1	10.2	9.2	8.3	7.2	6.2	5.1	3.9	2.7	1.4	0.0	1.4	2.8	4.2	5.6	7.0	8.3	9.6	10.8	12.0	13.2
0.9	11.2	10.3	9.4	8.4	7.4	6.3	5.2	4.0	2.8	1.4	0.0	1.5	2.9	4.4	5.8	7.2	8.6	9.9	11.2	12.4	13.6
1.0	11.3	10.4	9.5	8.6	7.6	6.5	5.4	4.1	2.9	1.5	0.0	1.5	3.0	4.5	6.0	7.5	8.9	10.2	11.5	12.8	14.0

Рассмотрим результаты, полученные при прохождении метриками всех четырех групп тестов.

Группа 1. Отсутствие одного символа в строке (или строки в тексте).

В данной группе тестов производится вычисление двух метрик – ранней базовой и текущей расширенной при сравнении строки «abcd» с набором строк, составленных из 1-й строки путем замены ее символов на «_». Количество всех таких комбинаций равно 16.

Результаты тестирования представлены в таблице 2.

Таблица 2

Результаты вычисления метрики_1 и метрики_2 для группы тестов 1 (при сравнении с «abcd»)

Table 2

Results of metrica_1 and metrica_2 calculation for test group 1 (when compared with "abcd")

№ теста	2-я строка	Метрика_1	Метрика_2
1	abcd	1.0000	1.0000
2	abc_	0.7500	0.7500
3	ab_d	0.7500	0.7500
4	a_cd	0.7500	0.7500
5	_bcd	0.7500	0.7500
6	ab__	0.5000	0.5000
7	a_c_	0.5000	0.5000
8	a__d	0.5000	0.5000
9	_bc_	0.5000	0.5000
10	_b_d	0.5000	0.5000
11	__cd	0.5000	0.5000
12	a___	0.2500	0.2500
13	_b__	0.2500	0.2500
14	__c_	0.2500	0.2500
15	___d	0.2500	0.2500
16	____	0.0000	0.0000

Результаты вычисления метрик согласно таблице 2 в виде гистограмм представлены на рисунке 1.

Анализ полученных результатов позволяет сделать следующие выводы.

Во-первых, значения метрик достаточно корректно отображают близость строк, поскольку тесты с бóльшим номером (соответствующим, в том числе и степени изменения первоначальной строки) имеют существенные отличия от строки «abcd».

Во-вторых, значения метрик идентичны, что вполне закономерно, поскольку в строках все символы или совпадают и находятся в одинаковых позициях, или отсутствуют (то есть заменены на «_»), что не требует учета коэффициента корректировки k^{String}.

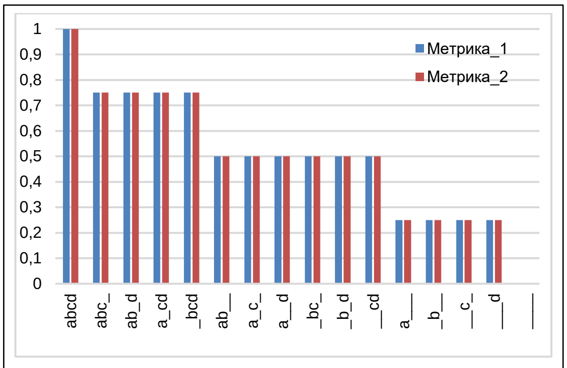

Рис. 1. Гистограмма результатов вычисления метрики_1 и метрики_2 для группы тестов 1 (при сравнении с «abcd»)

Fig. 1. Histogram of the calculation results of metrica_1 and metrica_2 for test group 1 (when compared with "abcd")

Группа 2. Присутствие одного символа в строке (или строки в тексте) в разных позициях.

В данной группе тестов производится вычисление метрик при сравнении 1-й строки «abcd» с набором строк, составленных из одного символа 1-й строки на разных позициях при заполнении остальных позиций символом «_». Количество таких комбинаций равно 16.

Результаты тестирования представлены в таблице 3.

Таблица 3

Результаты вычисления метрики_1 и метрики_2 для группы тестов 2 (при сравнении с «abcd»)

Table 3

Results of metrica_1 and metrica_2 calculation for test group 2 (when compared with "abcd")

№ теста	2-я строка	Метрика_1	Метрика_2
1	a___	0.2500	0.2500
2	_a__	0.1875	0.1867
3	__a_	0.1250	0.1229
4	___a	0.0625	0.0602
5	b___	0.1875	0.1867
6	_b__	0.2500	0.2500
7	__b_	0.1875	0.1852
8	___b	0.1250	0.1208
9	c___	0.1250	0.1229
10	_c__	0.1875	0.1852
11	__c_	0.2500	0.2500
12	___c	0.1875	0.1836
13	d___	0.0625	0.0602
14	_d__	0.1250	0.1208
15	__d_	0.1875	0.1836
16	___d	0.2500	0.2500

Результаты вычисления метрик согласно таблице 3 представлены в виде гистограмм на рисунке 2.

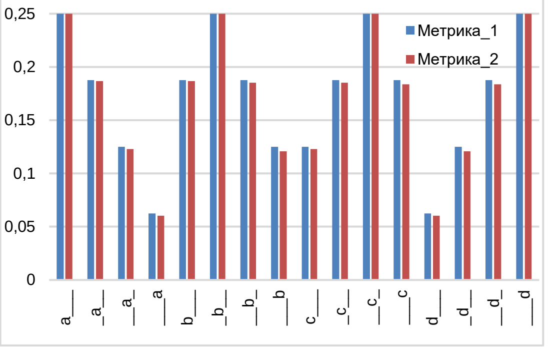

Рис. 2. Гистограмма результатов вычисления метрики_1 и метрики_2 для группы тестов 2 (для сравнения строк с «abcd»)

Fig. 2. Histogram of the calculation results of metrica_1 and metrica_2 for test group 2 (for comparing lines with “abcd”)

Анализ полученных результатов позволяет сделать следующие выводы.

Во-первых, значения метрик достаточно корректно отображают близость строк, поскольку перемещение символа из 1-й строки (то есть кроме заполнителя «_») от корректной позиции к удаленным приводит к существенному уменьшению близости строк, что для метрики_1 соответствует значениям 0.25, 0.1875, 0.1250 и 0.0625.

Во-вторых, значения метрики_2 незначительно отличаются от метрики_1 именно из-за учета позиции символа 2-й строки от ее начала. Так, смещение символа «a» вправо на 1 позицию дает значение 0.1867 вместо 0.1875, аналогичное смещение «b» – 0.1852, а «c» – 0.1836.

Группа 3. Комбинации символов строк (или строк текста).

В данной группе тестов производится вычисление метрик при сравнении 1-й строки «abcd» с набором строк, составленных из всех комбинаций ее символов (то есть без их дублирования и использования заполнителя «_»). Количество таких комбинаций равно 24, а их порядок такой, что вначале перебирается 4-й символ, а затем 3-й, 2-й и 1-й.

Результаты тестирования представлены в таблице 4.

Результаты вычисления метрик согласно таблице 4 представлены в виде гистограмм на рисунке 3.

Анализ полученных результатов позволяет сделать следующие выводы.

Таблица 4

Результаты вычисления метрики_1 и метрики_2 для группы тестов 3 (при сравнении с «abcd»)

Table 4

Results of metrica_1 and metrica_2 calculation for test group 3 (when compared with “abcd”)

№ теста	2-я строка	Метрика_1	Метрика_2
1	abcd	1.0000	1.0000
2	abdc	0.8750	0.8672
3	acbd	0.8750	0.8703
4	acdb	0.7500	0.7396
5	adbc	0.7500	0.7396
6	adcb	0.7500	0.7417
7	bacd	0.8750	0.8734
8	badc	0.7500	0.7406
9	bcad	0.7500	0.7448
10	bcda	0.6250	0.6156
11	bdac	0.6250	0.6141
12	bdca	0.6250	0.6177
13	cabd	0.7500	0.7448
14	cadb	0.6250	0.6141
15	cbad	0.7500	0.7458
16	cbda	0.6250	0.6167
17	cdab	0.5000	0.4875
18	cdba	0.5000	0.4891
19	dabc	0.6250	0.6156
20	dacb	0.6250	0.6177
21	dbac	0.6250	0.6167
22	dbca	0.6250	0.6203
23	dcab	0.5000	0.4891
24	dcba	0.5000	0.4906

Во-первых, метрики достаточно адекватно учитывают близость строк, что индицируется постепенным уменьшением ее значения для

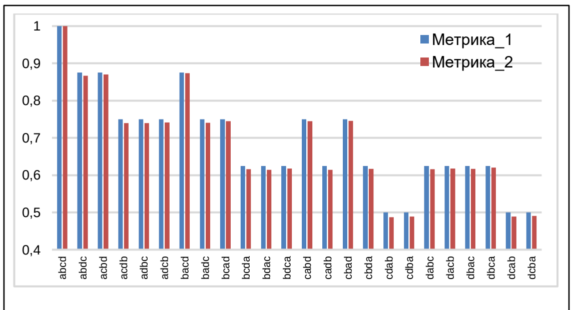

Рис. 3. Гистограмма результатов вычисления метрики_1 и метрики_2 для группы тестов 3 (для сравнения строк с «abcd»)

Fig. 3. Histogram of calculation results of metrica_1 and metrica_2 for test group 3 (for comparing lines with “abcd”)

последующих комбинаций символов во 2-й строке.

Во-вторых, зависимость между значением метрик и комбинациями символов 2-й строки существенно сложнее, чем простой снижающийся тренд, что также закономерно. Так, 7-й тест («bacd») обладает большей близостью, чем 6-й («adcb»), имеющий совпадение 1-го символа «a» (с 1-й строкой «abcd»). Это можно обосновать интегральностью вычисления метрик, складывающейся из близости всех четырех символов строк.

В-третьих, незначительное отличие значений метрики_2 от метрики_1, как и ранее, обосновано учетом позиции символов (точнее их середины). Так, например, перестановка двух ближайших символов во 2-й строке дает одинаковое значение метрики_1 – 0.8750, хотя метрика_2 учитывает их позицию, уменьшая свое значение для более отдаленных: «bacd» – 0.8734 (тест 7), «acbd» – 0.8703 (тест 3), «abdc» – 0.8672 (тест 2).

Группа 4. Изменение количества символов строки (или строк текста).

В данной группе тестов производится вычисление двух метрик при сравнении 1-й строки «12345» с набором строк, составленных последовательным увеличением ее длины сначала совпадающими символами из 1-й строки, то есть до 5, а затем, добавляя новые символы, до 20. Количество таких комбинаций равно 21, поскольку 1-й тест содержит пустую строку.

Итоги тестирования представлены в таблице 5.

Результаты вычисления метрик согласно таблице 5 представлены в виде гистограмм на рисунке 4.

Анализ полученных результатов позволяет сделать следующие выводы:

– по мере дополнения 2-й строки до 1-й (тесты с 1-го по 6-й) значение метрик линейно растет от 0 до 1 с шагом 0.2, что вполне корректно и точно отражает их близость;

– по мере дополнения 2-й строки новыми символами (тесты с 7-го по 21-й) значение метрик снижается (с окололографмической, но не являющейся таковой, закономерностью), что позволяет с достаточной точностью как отражать совпадение начальной части 2-й строки с 1-й, так и учитывать наличие в ней новых символов, отсутствующих в 1-й;

– значения метрик идентичны, что вполне закономерно, поскольку 2-я строка содержит часть символов 1-й строки на корректных позициях, тождественна ей или же дополнена новыми символами.

Таблица 5

Результаты вычисления метрики_1 и метрики_2 для группы тестов 4 (при сравнении с «12345»)

Table 5

Results of metrica_1 and metrica_2 calculation for test group 4 (when compared with “12345”)

№ теста	2-я строка	Метрика_1	Метрика_2
1		0.0000	0.0000
2	1	0.2000	0.2000
3	12	0.4000	0.4000
4	123	0.6000	0.6000
5	1234	0.8000	0.8000
6	12345	1.0000	1.0000
7	123456	0.8333	0.8333
8	1234567	0.7143	0.7143
9	12345678	0.6250	0.6250
10	123456789	0.5556	0.5556
11	1234567890	0.5000	0.5000
12	1234567890a	0.4545	0.4545
13	1234567890ab	0.4167	0.4167
14	1234567890abc	0.3846	0.3846
15	1234567890abcd	0.3571	0.3571
16	1234567890abcde	0.3333	0.3333
17	1234567890abcdef	0.3125	0.3125
18	1234567890abcdefg	0.2941	0.2941
19	1234567890abcdefgh	0.2778	0.2778
20	1234567890abcdefghi	0.2632	0.2632
21	1234567890abcdefghik	0.2500	0.2500

Обсуждение результатов

Укажем ряд особенностей текущей версии метрики, которые, хотя в ряде случаев и могут считаться недостатками, не являются критичными и позволяют применять ее в интересах генетической деэволюции.

Во-первых, вводимые коэффициенты дополнительной корректировки близости (то есть k^{Text} и k^{String}) при различных позициях символов (и строк) оказывают не такое существенное влияние, как предполагалось изначально (табл. 1). Следовательно, требуется дополнительное исследование. Впрочем, для генетической деэволюции даже незначительный учет дальности символов и строк может иметь существенное значение для решения задачи подбора исходного кода, поскольку оно качественно повлияет на время и успешность селекции.

Во-вторых, несмотря на работоспособность метрики и ее применимость в алгоритмах гене-

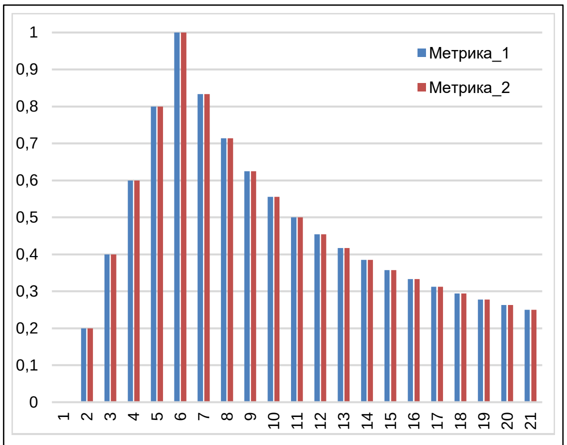

Рис. 4. Гистограмма результатов вычисления метрики_1 и метрики_2 для группы тестов 4 (при сравнении с «12345»)

Fig. 4. Histogram of the calculation results of metrica_1 and metrica_2 for test group 4 (when compared with "12345")

тической деэволюции, она не позволяет в полной мере учесть конкретный формальный синтаксис ассемблерного кода программы, описывая его как список строк.

В-третьих, были рассмотрены лишь достаточно общие случаи сравнения текстов и строк, что не позволяет оценивать в полной мере работоспособность метрики для более сложных случаев. Тем не менее такая ситуация в принципе не имеет однозначного решения, что не позволяет требовать его и от метрики.

Заключение

Результатом исследования стала новая авторская версия метрики близости двух программ, представленных в виде текстов ассемблерного кода программы, которая отличается от предыдущей версии бóльшей чувствительностью к различиям как в текстовых строках, так и в их символах (путем учета позиции конструкций относительно начала). Аналитическая запись метрики, а также ряд преимуществ (например, инвариантность к синтаксису текстов, оценочная шкала на отрезке [0 …1] и др.) позволяют применять ее как часть комплексных математических инструментов без дополнительных доработок. Проведенные над метрикой эксперименты говорят не только о ее работоспособности в интересах генетической деэволюции, но и об общей востребованности в сфере информационных технологий и их безопасности.

Продолжением исследования должны стать устранение указанных недостатков метрики, а также синтез ее следующей версии, еще более инвариантной к сравниваемым представлениям данных за счет использования формальных синтаксисов этих представлений, передаваемых в метрику как один из параметров.

Список литературы

1. Гусева Н.И. Пространства над алгебрами с евклидовой метрикой // Итоги науки и техники. Сер. Современная математика и ее приложения. Тематические обзоры. 2020. Т. 179. С. 10–15. doi: 10.36535/0233-6723-2020-179-10-15.
2. Израилов К.Е., Гололобов Н.В., Краскин Г.А. Метод анализа вредоносного программного обеспечения на базе Fuzzy Hash // Информатизация и связь. 2019. № 2. С. 36–44.
3. Luo L., Ming J., Wu D., Liu P., Zhu S. Semantics-based obfuscation-resilient binary code sim-ilarity comparison with applications to software and algorithm plagiarism detection. IEEE Transactions on Software Engineering, 2017, vol. 43, no. 12, pp. 1157–1177. doi: 10.1109/TSE.2017.2655046.
4. Ашуров А.Я. Реализация переполнения стекового буфера программ и методы противодействия уязвимости // Системы синхронизации, формирования и обработки сигналов. 2023. Т. 14. № 1. С. 10–17.
5. Liu S., Yu C., Wang L., Wang C., Xu G. FF-BCSD: A Binary code similarity detection method based on feature fusion. Proc. ICCC, 2023, pp. 757–761. doi: 10.1109/ICCC59590.2023.10507703.
6. Ghosh K., Otero D. Discovering authorship of vulnerabilities in open source software. Proc. APSEC Workshops, 2021, pp. 41–46. doi: 10.1109/APSECW53869.2021.00018.
7. Израилов К.Е. Концепция генетической деэволюции представлений программы. Ч. 1 // Вопросы кибербезопасности. 2024. № 1. С. 61–66. doi: 10.21681/2311-3456-2024-1-61-66.
8. Израилов К.Е. Концепция генетической деэволюции представлений программы. Ч. 2 // Вопросы кибербезопасности. 2024. № 2. С. 81–86. doi: 10.21681/2311-3456-2024-2-81-86.
9. Fletcher S., Islam M.Z. Comparing sets of patterns with the Jaccard index. Australasian J. of Information Systems, 2018, vol. 22. doi: 10.3127/ajis.v22i0.1538.
10. Татарникова Т.М., Богданов П.Ю. Построение психологического портрета человека с применением технологий обработки естественного языка // Науч.-технич. вестн. информационных технологий, механики и оптики. 2021. Т. 21. № 1. С. 85–91. doi: 10.17586/2226-1494-2021-21-1-85-91.
11. Hu Y., Wang H., Zhang Y., Li B., Gu D. A semantics-based hybrid approach on binary code similarity comparison. IEEE Transactions on Software Engineering, 2021, vol. 47, no. 6, pp. 1241–1258. doi: 10.1109/TSE.2019.2918326.
12. Усачев Ю.Е. Вычисление степени семантической близости документов // XXI век: итоги прошлого и проблемы настоящего плюс. 2016. № 6. С. 96–103.

13. Ермолаев А.В. Методы анализа и визуализации структуры данных о близости // Социология: 4М. 2005. № 21. С. 150–171.

14. Naveena P., Rao P.K.S. Detection of near duplicates over graph datasets using pruning. Proc. INDISCON, 2020, pp. 309–313. doi: 10.1109/INDISCON50162.2020.00068.

15. Аникин И.В., Исяндавлетова Я.М. Реверсивный анализ вредоносного программного обеспечения Raccoon Stealer // Инженерный вестник Дона. 2023. № 4. С 238–251.

Software & Systems doi: 10.15827/0236-235X.149.089-099 2025, 38(1), pp. 89–99

Author's metric for assessing proximity of programs: Application for vulnerability search using genetic de-evolution

Mikhail V. Buynevich [1], Konstantin E. Izrailov [2]✉

[1] St. Petersburg University of State Fire Service of EMERCOM of Russia, St. Petersburg, 196105, Russian Federation
[2] St. Petersburg Federal Research Center of RAS, St. Petersbur, 199178, Russian Federation

For citation
Buynevich, M.V., Izrailov, K.E. (2025) ‘Author's metric for assessing proximity of programs: Application for vulnerability search using genetic de-evolution’, *Software & Systems*, 38(1), pp. 89–99 (in Russ.). doi: 10.15827/0236-235X.149.089-099
Article info
Received: 10.06.2024 After revision: 25.06.2024 Accepted: 28.06.2024

Abstract. The paper is relevant due to the tasks in the field of information security that require comparison of programs in their different representations, for example, in text assembly code (e.g., for vulnerability search or authorship verification). The paper presents a proximity metric for two texts in the form of a character string list, which is a development of its previous author's version. The main result of the current study (as a part of the main study aimed at genetic de-evolution of programs) is the metric itself, as well as its characteristics and peculiarities revealed through experiments. The paper presents the metric in analytical form implemented in Python. The metric takes at the input two lists of character lines for comparison, and the coefficients of taking into account the element position from the beginning of the list and the character sequence. The calculation result is a numeric value in the range from 0 to 1. Metric's novelty is in a sufficiently accurate and sensitive assessment of two texts' proximity regardless of data representation formats. The current metric version differs from the previous one by taking into account the mentioned coefficients. Theoretical significance lies in the development of comparing methods for arbitrary texts that are a list of character lines containing information, which appears sequentially according to a certain logic (requires position consideration). Besides the general purpose of comparative tools like this, the metric is practically relevant due to the possibility of determining the proximity of two programs. These programs have a binary representation of the machine code. It is pre-transformed into a textual representation of an assembly code.
Keywords: information security, proximity metric, software, vulnerability search, basic principle, genetic de-evolution
Acknowledgements. The work was partially financially supported from budget topic FFZF-2025-0016

References

1. Guseva, N.I. (2020) ‘Spaces over algebras with Euclidean metric’, *Progress in Sci. and Tech. Ser. Contemporary Mathematics and Its Applications. Thematic Surveys*, 179, pp. 10–15 (in Russ.). doi: 10.36535/0233-6723-2020-179-10-15.

2. Izrailov, K.E., Gololobov, N.V., Kraskin, G.A. (2019) ‘Method of malware analysis based on Fuzzy Hash’, *Informatization and Communication*, (2), pp. 36–44 (in Russ.).

3. Luo, L., Ming, J., Wu, D., Liu, P., Zhu, S. (2017) ‘Semantics-based obfuscation-resilient binary code similarity comparison with applications to software and algorithm plagiarism detection’, *IEEE Transactions on Software Engineering*, 43(12), pp. 1157–1177. doi: 10.1109/TSE.2017.2655046.

4. Ashurov, A.Ya. (2023) ‘Stack buffer overflow of programs implementation and methods of coun-teracting the vulnerability’, *Systems of Signal Synchronization, Generating and Processing*, 14(1), pp. 10–17 (in Russ.).

5. Liu, S., Yu, C., Wang, L., Wang, C., Xu, G. (2023) ‘FF-BCSD: A Binary code similarity detection method based on feature fusion’, *Proc. ICCC*, pp. 757–761. doi: 10.1109/ICCC59590.2023.10507703.

6. Ghosh, K., Otero, D. (2021) ‘Discovering authorship of vulnerabilities in open source software’, *Proc. APSEC Workshops*, pp. 41–46. doi: 10.1109/APSECW53869.2021.00018.

7. Izrailov, K.E. (2024) 'The genetic de-evolution concept of program representations. Pt 1', *Cybersecurity Issues*, (1), pp. 61–66 (in Russ.). doi: 10.21681/2311-3456-2024-1-61-66.
8. Izrailov, K.E. (2024) 'The genetic de-evolution concept of program representations. Pt 2', *Cybersecurity Issues*, (2), pp. 81–86 (in Russ.). doi: 10.21681/2311-3456-2024-2-81-86.
9. Fletcher, S., Islam, M.Z. (2018) 'Comparing sets of patterns with the Jaccard index', *Australasian J. of Information Systems*, 22. doi: 10.3127/ajis.v22i0.1538.
10. Tatarnikova, T.M., Bogdanov, P.Yu. (2021) 'Human psyche creation by application of natural language processing technologies', *Sci.Tech. J. Inf. Technol. Mech. Opt.*, 21(1), pp. 85–91 (in Russ.). doi: 10.17586/2226-1494-2021-21-1-85-91.
11. Hu, Y., Wang, H., Zhang, Y., Li, B., Gu, D. (2021) 'A semantics-based hybrid approach on binary code similarity comparison', *IEEE Transactions on Software Engineering*, 47(6), pp. 1241–1258. doi: 10.1109/TSE.2019.2918326.
12. Usachev, Yu.E. (2016) 'The calculation of the degree of semantic proximity of documents', *XXI Century: Resumes of the Past and Challenges of the Present Plus*, (6), pp. 96–103 (in Russ.).
13. Ermolaev, A.V. (2005) 'Methods for analyzing and visualizing the structure of proximity data', *Sociology: 4M*, (21), pp. 150–171 (in Russ.).
14. Naveena, P., Rao, P.K.S. (2020) 'Detection of near duplicates over graph datasets using pruning', *Proc. INDISCON*, pp. 309–313. doi: 10.1109/INDISCON50162.2020.00068.
15. Anikin, I.V., Isyandavletova, Ya.M. (2023) 'Reverse analysis of malware Raccoon Stealer', *Ing. J. of Don*, (4), pp. 238–251 (in Russ.).

Авторы
Буйневич Михаил Викторович [1],
д.т.н., профессор, профессор кафедры,
bmv1958@yandex.ru
Израилов Константин Евгеньевич [2],
к.т.н., доцент, старший научный сотрудник,
konstantin.izrailov@mail.ru

[1] Санкт-Петербургский университет Государственной противопожарной службы МЧС России,
г. Санкт-Петербург, 196105, Россия
[2] Санкт-Петербургский Федеральный исследовательский центр РАН,
г. Санкт-Петербург, 199178, Россия

Authors
Mikhail V. Buynevich [1], Dr.Sci. (Engineering),
Professor, Professor of Department,
bmv1958@yandex.ru
Konstantin E. Izrailov [2], Cand. of Sci. (Engineering),
Associate Professor, Senior Researcher,
konstantin.izrailov@mail.ru

[1] St. Petersburg University of State Fire Service of EMERCOM of Russia, St. Petersburg,
196105, Russian Federation
[2] St. Petersburg Federal Research Center of RAS,
St. Petersbur, 199178,
Russian Federation