ТЕХНОЛОГИЧЕСКИЕ ОСНОВЫ ПОВЫШЕНИЯ НАДЕЖНОСТИ И КАЧЕСТВА В ПРИБОРОСТРОЕНИИ И РАДИОЭЛЕКТРОНИКЕ

TECHNOLOGICAL BASICS FOR IMPROVING RELIABILITY AND QUALITY IN INSTRUMENTATION AND RADIO ELECTRONICS

УДК 621.317
doi:10.21685/2307-4205-2022-3-11

ПРОТИВОДЕЙСТВИЕ АТАКАМ МАРШАЛКО НА СЕТИ ИСКУССТВЕННЫХ НЕЙРОНОВ ЗА СЧЕТ ВВЕДЕНИЯ ЛОЖНЫХ СВЯЗЕЙ

И. А. Крохин[1], М. Ю. Михеев[2]

[1, 2] Пензенский государственный технологический университет, Пенза, Россия
[1] kr.ig.sv08@gmail.com, [2] mix1959@gmail.com

Аннотация. Целью работы является добавление ПО эмулятора нейросети и маскирующей части нейрона в процесс обучения нейронной сети по стандарту ГОСТ Р52633 для систем аутентификации пользователя на предмет возможности повышения защиты личных криптографических ключей пользователя при атаке Маршалко.

Ключевые слова: эмулятор нейросети, криптографический ключ, атака Маршалко, обучение нейросети, аутентификация пользователя

Для цитирования: Крохин И. А., Михеев М. Ю. Противодействие атакам Маршалко на сети искусственных нейронов за счет введения ложных связей // Надежность и качество сложных систем. 2022. № 3. С. 86–94. doi:10.21685/2307-4205-2022-3-11

COUNTERING MARSHALKO'S ATTACKS ON ARTIFICIAL NEURON NETWORKS BY INTRODUCING FALSE CONNECTIONS

I.A. Krohin[1], M.Yu. Mikheev[2]

[1, 2] Penza State Technological University, Penza, Russia
[1] kr.ig.sv08@gmail.com, [2] mix1959@gmail.com

Abstract. The aim of the work is to add the neural network emulator software and the masking part of the neuron to the neural network training process according to GOST R52633 standard for user authentication systems, for the possibility of increasing the protection of the user's personal cryptographic keys in the event of a Marshalko attack.

Keywords: neural network emulator, cryptographic key, Marshalko attack, neural network training, user authentication

For citation: Krokhin I.A., Mikheev M.Yu. Countering Marshalko's attacks on artificial neuron networks by introducing false connections. *Nadezhnost' i kachestvo slozhnykh sistem = Reliability and quality of complex systems.* 2022;(3):86–94. (In Russ.). doi:10.21685/2307-4205-2022-3-11

Из-за масштабной информатизации современного общества возникает необходимость обеспечения высоконадежной идентификации личности пользователя в информационном пространстве. Одним из наиболее перспективных направлений защиты личных данных является биометрическое распознавание пользователя [1].

Программно-аппаратное решение

Атака Маршалко строится на наблюдении большого числа выходов у незащищенных нейронов и поиска общих связей [2, 3].

Одним из возможных программно-аппаратных решений противодействия атаке Маршалко является использование ПО эмулятора нейросети. Все вычисления будут находиться в условном «Черном ящике» для атакующего систему биометрической идентификации. Соответственно, атакующий не сможет узнать весовые коэффициенты нейронов, так как они будут закрыты. Поэтому необходимо реализовать обучение нейронной сети, только в ПО эмулятора нейросети (рис. 1).

Рис. 1. Схема реализации ПО эмулятора нейросети

С использованием ПО эмулятора нейросети для атаки Маршалко необходимо изначально узнать, какие входные биометрические параметры влияют на какие входы нейронов. Для этого после имитации обучения (расчета весовых коэффициентов нейронов) использовался генератор входных биометрических параметров типа «Чужой», которые подавались на вход функции имитирующую работу нейронной сети. Функция, написанная на языке C++, представлена ниже:

```
bool* work_neuron(double input_param[NUM_BIO_PARAM])
{
  bool result_output[NUM_NEURON];
  double res_weight = 0;
  int num_input_param = 0;
  for (int i = 0; i < NUM_NEURON; i++) // прогон по всем нейронам
  {
for (int weight = 1; weight < NUM_INPUT_NEURON; weight++)
// прогон по всем весам
    {
      num_input_param = Neural_network.Neuron[i].sootvetstvie[weight - 1];
      res_weight += (Neural_network.Neuron[i].weight[weight] *
input_param[num_input_param]) - Neural_network.Neuron[i].weight[0];
    }

    if (res_weight >= Neural_network.Neuron[i].porog) // Пороговая функция
      result_output[i] = 1;
    else    result_output[i] = 0;
    res_weight = 0;
  }
  return result_output;
}
```

Расчет выходного бита нейрона осуществляется с помощью пороговой функции значения вычисляемого по формуле

$$Y = \sum_{i=1}^{n} \mu_i X_i - \mu_0 \ ,$$

где μ – весовой коэффициент; X – входной биометрический параметр; n – число входов нейрона.

Для примера поиска взаимосвязи входов биометрических параметров с входами нейронов был выбран диапазон биометрических параметров от –200 до 200 с шагом 25, и вход биометрических параметров № 215.

Информация о соответствии входа биометрических параметров № 215, находящаяся в ПО эмулятора нейросети, представлена в табл. 1.

Таблица 1

Соответствие входа биометрических параметров № 215 к входам нейронов нейронной сети

Номер нейрона	Номер входа нейрона	Весовой коэффициент
13	8	0,015795
57	4	0,028119
73	10	0,033062
186	11	-0,125872

Результат выполнения программы представлен в табл. 2.

Таблица 2

Результат выполнения программы по поиску соответствий входа биометрических параметров с входами нейронов

Биометрич. параметры	–200	–175	–150	–125	–100	–75	–50	–25
№ 13	0	0	0	0	0	0	0	0
№ 57	0	0	0	0	0	0	0	0
№ 73	0	0	0	0	0	0	0	0
№ 186	1	1	1	1	1	1	1	1
	0	25	50	75	100	125	150	175
№ 13	0	0	0	1	1	1	1	1
№ 57	0	0	0	0	0	0	0	0
№ 73	0	0	0	1	1	1	1	1
№ 186	0	0	0	0	0	0	0	0

В таблице представлены выходные биты нейронов от различных биометрических параметров. Из таблицы видно, что чем больше весовой коэффициент нейрона и входной биометрический параметр, тем раньше начинает дрожать выходной бит нейрона. Так же из таблицы видно, что нейрон № 57 так и не изменил выходного бита.

С использованием ПО эмулятора нейросети для работы с нейронной сетью биометрической аутентификации, увеличивается сложность проведения атак Маршалко, так как необходимо изначально проверить каждый вход биометрических параметров на соответствие входам нейронов. А так же данная проверка не может гарантировать нахождение всех соответствий, например из-за маленьких весовых коэффициентов.

Расчет весовых коэффициентов реальных связей по ГОСТ Р 52633

Алгоритм обучения нейронной сети биометрической идентификации по ГОСТ Р 52633.5[1]:

– для $j = 1,\ldots,\left[\frac{N_0}{n}\right]$ и $i = 1,\ldots,n$, где N_0 – число входов нейронной сети, n – число входов каждого из нейронов.

[1] ГОСТ Р 52633.5–2011. Защита информации. Техника защиты информации. Автоматическое обучение нейросетевых преобразователей биометрии.

Рассчитываем весовые коэффициенты по формуле

$$\mu_i^j = \frac{(-1)^{b_j}\left(\overline{v_{jn+i}^1} - \overline{v_{jn+i}^0}\right)}{a^2\sqrt{S^2\left(v_{jn+i}^1\right)} + \sqrt{S^2\left(v_{jn+i}^0\right)}},$$

где $\overline{v_t^c}$ – выборочное среднее; $S^2\left(v_t^c\right)$ – дисперсия *t*-го биометрического параметра $t = 1,\ldots,v_{N_0}$ для образов «Свой» ($c = 0$) и «Чужой» ($c = 1$).

Заполняем таблицу соответствий по формуле

$$d_i^j = jn + i;$$

– для $j = \left[\frac{N_0}{n}\right] + 1,\ldots,N_1$ и $i = 1,\ldots,n$, где N_1 – число нейронов в сети.

Рассчитываем весовые коэффициенты по формуле

$$\mu_i^j = \frac{(-1)^{b_j}\left(\overline{v_\tau^1} - \overline{v_\tau^0}\right)}{a^2\sqrt{S^2\left(v_\tau^1\right)} + \sqrt{S^2\left(v_\tau^0\right)}},$$

где τ – псевдолучайная выбранная координата вектора биометрических параметров;

– для $j = 1,\ldots,N_1$. Вычисляем значение порогового элемента нейрона по формуле

$$\mu_0^j = norm\left(\mu_1^j,\ldots,\mu_n^j\right).$$

По вышеописанному алгоритму была реализована программа на языке программирования C++ расчета весовых коэффициентов реальных связей:

```
int calc_weight(){
 if ((Input_bio_param.Chyzhoy == NULL) || (Input_bio_param.Svoy == NULL)) return -1;
 int num = 0;
 double mean_svoy = 0, mean_chyzhoy = 0, disp_svoy = 0, disp_chyzhoy = 0;
// Заполняем весовые коэффициенты нейронов
 for (int j = 0; j < NUM_NEURON; j++){
  for (int i = 1; i < NUM_INPUT_NEURON; i++){
   if (j < (NUM_BIO_PARAM / NUM_INPUT_NEURON))
// Заполняем веса для нейронов от 1 до (N0/n)
    num = (j * NUM_INPUT_NEURON) + i;
   else
// Заполняем веса для нейронов от (N0/n + 1) до N1
    num = get_rand_range_int(1, NUM_BIO_PARAM) - 1;
   mean_chyzhoy = calc_mean(num, Input_bio_param.Chyzhoy);
// Расчитываем среднее значение типа "Чужой"
   mean_svoy = calc_mean(num, Input_bio_param.Svoy);
// Расчитываем среднее значение типа "Свой"
   disp_chyzhoy = calc_disp(num, Input_bio_param.Chyzhoy, mean_chyzhoy);
// Расчитываем дисперсию типа "Чужой"
   disp_svoy = calc_disp(num, Input_bio_param.Svoy, mean_svoy);
// Расчитываем дисперсию типа "Свой"
   if ((disp_chyzhoy == 0) && (disp_svoy == 0))
    Neural_network.Neuron[j].weight[i] = 0;
   else
    Neural_network.Neuron[j].weight[i] = (mean_chyzhoy - mean_svoy) /
(sqrt(disp_chyzhoy) + sqrt(disp_svoy)); // Записываем вес m(ji)
   Neural_network.Neuron[j].sootvetstvie[i] = num; // Записываем соответствие d(ji)
   }
  Neural_network.Neuron[j].num = j; // Записываем порядковый номер нейрона.
  }
// Заполняем пороговые значения нейронов
 for (int j = 0; j < NUM_NEURON; j++){
  Neural_network.Neuron[j].weight[0] = norm(Neural_network.Neuron[j].weight);
  Neural_network.Neuron[j].porog = Neural_network.Neuron[j].weight[0];
 }
 return 0;
}
```

Реализация атаки Маршалко

Атака Маршалко строится на наблюдении большого числа выходов у незащищенных нейронов и поиска общих связей [2]. Схема нахождения общих связей нейронов при обучении по ГОСТ Р52633.5 представлена на рис. 2.

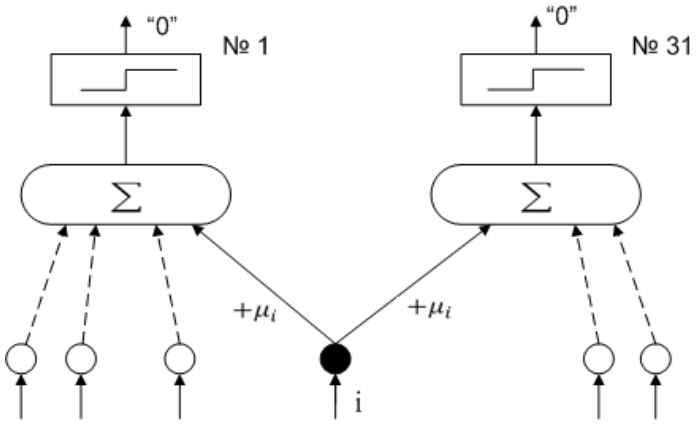

Рис. 2. Схема нахождения общих связей

Реализация атаки, т.е. поиск общих связей у нейронов, представлена на языке программирования C++:

```
int attack()
{
 int level_link = 0;
 double weight[NUM_INPUT_NEURON];
 for (int num_neuron1 = 0; num_neuron1 < NUM_NEURON - 1; num_neuron1++)
 {
// Проверка, чтобы первый сравниваемый нейрон не состоял в группах
  if (Neural_network.Neuron[num_neuron1].attack_group == 0){
   for (int num_neuron2 = num_neuron1 + 1; num_neuron2 < NUM_NEURON; num_neuron2++){
// Проверка, чтобы второй сравниваемый нейрон не состоял в группах
    if (Neural_network.Neuron[num_neuron2].attack_group != 0) break;
// Производим сравнение двух нейронов не состоящих в группах на текущий момент
    level_link = Search_link(num_neuron1, num_neuron2, weight);
    if (level_link != 0) { //Если есть общая связь
     Neural_network.cur_group++; // Увеличиваем индекс текущей группы
     Write_info_link(num_neuron1, num_neuron2, level_link, weight);
// Записываем информацию о связи
     Check_group(); // Проверяем все оставшиеся нейроны на соответствие текущей группы
     }
    }
   }
  }
  Searh_free_link(); // Поиск и запись в структуру свободных нейронов
  return 0;
}
```

Алгоритм поиска общих связей заключается в сравнении по модулю весовых коэффициентов нейронов в функции Search_link. При нахождении общей связи нейронам присваивается группа, затем происходит поиск всех оставшихся нейронов, не состоящих в группах на соответствии текущей группе нейронов с общими связями, функцией Check_group. Результат работы программы по поиску общих связей у 256 линейных нейронов с 16 входами представлен на рис. 3.

Исходя из результата, можно сделать вывод:

– свободные нейроны без общих связей отсутствуют;

– все нейроны находятся в первой группе;

– общие связи между нейронами распределены в следующем порядке: 188 по одному входу (одинарная связь), 56 по два входа (двойная связь), 7 по три входа, 3 по четыре входа и 1 связь имеет 5 входов нейронов одновременно.

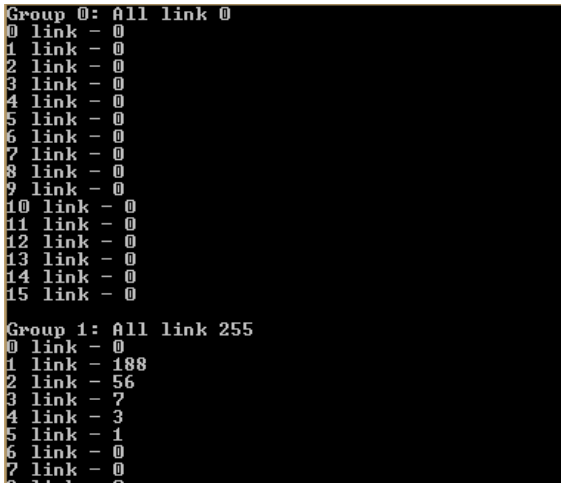

Рис. 3. Результат выполнения программы поиска общих связей

В табл. 3 представлены весовые коэффициенты двух случайных нейронов.

Таблица 3

Весовые коэффициенты двух случайных нейронов

Номер нейрона	1	2	3	4	5	6	7	8
39	0,17732	0,01595	0,11111	0,01381	0,06010	0,04882	0,03137	**0,03749**
44	0,13645	0,03604	0,03837	0,03885	0,01462	**0,03341**	0,02529	**0,03749**
	9	10	11	12	13	14	15	16
39	0,01490	0,02484	0,03365	0,07057	0,03012	**0,03341**	0,02534	0,02555
44	0,07456	0,02160	0,00300	0,04203	0,01645	0,03182	0,03530	0,02512

Как видно из таблицы, у нейронов общая связь по двум входам, т.е. двойная общая связь.

Противодействие изучению связей нейронной сети маскированием весовых коэффициентов

Одним из возможных программных решений противодействия атаке Маршалко является выполнение итерационного дообучения нейронной сети биометрической идентификации с целью маскирования значимых весовых коэффициентов [3]. Необходимо увеличить количество весовых коэффициентов нейрона в два раза, условно до 32, тогда атакующий не сможет точно идентифицировать общие связи значимых нейронов. Схема нейрона с использованием маскирующей части представлена на рис. 4.

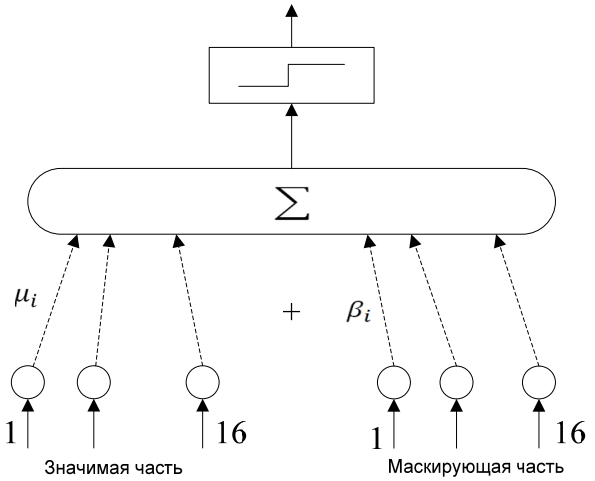

Рис. 4. Схема нейрона с использованием маскирующей части

Добавление маскирующих весовых коэффициентов реализовано на языке программирования C++:

```
int add_mask_elements()
{
 int Neuron1_ID = 0, Neuron2_ID = 0;
 double weight = 0;
 for (int group_id = 1; group_id < Neural_network.cur_group + 1; group_id++) {
// Анализируем все заполненные группы
  for (int link_id = 0; link_id < Neural_network.group[group_id].sum_link; link_id++){
// Анализируем все общие связи в группе
   for (int level_id = 0; level_id < Neural_network.group[group_id].link[link_id].
level_link; level_id++) {
// Анализируем все уровни
    Neuron1_ID = Neural_network.group[group_id].link[link_id].neuron1;
    Neuron2_ID = Neural_network.group[group_id].link[link_id].neuron2;
    weight = Neural_network.group[group_id].link[link_id].weight[level_id];

    for (int in_neuron_id = 0; in_neuron_id < NUM_INPUT_NEURON; in_neuron_id ++) {
// Проверка первого нейрона, на наличие такого веса в маске
     if (fabs(Neural_network.Neuron[Neuron1_ID].mask_weight[in_neuron_id]) ==
fabs(weight)) break; // Сравниваем веса по модулю
     else if ((in_neuron_id == NUM_INPUT_NEURON - 1)&&(Neural_network.
Neuron[Neuron1_ID].cur_mask_weight < NUM_INPUT_NEURON)){
Neural_network.Neuron[Neuron1_ID].mask_weight[Neural_network.Neuron[Neuron1_ID].
cur_mask_weight] = weight * (-1); // Добавляем вес в маску с обратным знаком
Neural_network.Neuron[Neuron1_ID].cur_mask_weight++;
     }
    }
    for (int in_neuron_id = 0; in_neuron_id < NUM_INPUT_NEURON; in_neuron_id++) {
// Проверка второго нейрона, на наличие такого веса в маске
     if (fabs(Neural_network.Neuron[Neuron2_ID].mask_weight[in_neuron_id]) ==
fabs(weight)) break;// Сравниваем веса по модулю
     else if ((in_neuron_id == NUM_INPUT_NEURON - 1) && (Neu-
ral_network.Neuron[Neuron2_ID].cur_mask_weight < NUM_INPUT_NEURON)){
Neural_network.Neuron[Neuron2_ID].mask_weight[Neural_network.Neuron[Neuron2_ID].
cur_mask_weight] = weight * (-1); // Добавляем вес в маску с обратным знаком
Neural_network.Neuron[Neuron2_ID].cur_mask_weight++;
     }
    }
   }
  }
 }
 return 0;
}
```

Алгоритм маскирования заключается в добавлении весового коэффициента в маскирующую часть нейрона с обратным знаком при обнаружении общей связи между нейронами.

В табл. 4 представлен пример заполнения маскирующими весовыми коэффициентами 39 и 44 нейронов.

Таблица 4

Маскирующие весовые коэффициенты

Номер нейрона	1	2	3	4	5	6	7	8
39	–0,02534	–0,03749	–0,03341	–0,01595	–0,01490			
44	–0,03749	–0,03341						
	9	10	11	12	13	14	15	16
39								
44								

Затем необходимо заполнить оставшиеся маскирующие входы нейронов. Также нужно не ухудшить качество биометрической идентификации пользователей, для этого весовые коэффициенты маскирующей части нейрона в сумме должны равняться нулю.

Генерация оставшихся весовых коэффициентов маскирующей части нейрона представлена ниже:

```
void generate_free_mask_weight()
{
 double need_sum = 0; // необходимая сумма генерируемых весов
 int num_weight = 0; // число генерируемых весов
 for (int num_neuron = 0; num_neuron < NUM_NEURON; num_neuron++) {
  num_weight = NUM_INPUT_NEURON - Neu-
ral_network.Neuron[num_neuron].cur_mask_weight;
  need_sum = 0;
  for (int num_weight_id = 0; num_weight_id < Neu-
ral_network.Neuron[num_neuron].cur_mask_weight; num_weight_id++){
   need_sum += Neural_network.Neuron[num_neuron].mask_weight[num_weight_id];
// Подсчитываем сумму текущих весов
  }
  need_sum *= (-1); // меняем знак, чтобы итоговая сумма равнялась 0
  generate_weight(num_neuron, num_weight, need_sum);
 }
}
```

Функция generate_weight является генератором псевдослучайных величин в диапазоне –0,3 до 0,3. Последний весовой коэффициент маскирующей части нейрона является регулирующим, чтобы выполнялось условие $\sum_{1}^{16}\beta_i * X \approx 0$.

В табл. 5 представлены маскирующие весовые коэффициенты нейронов 39 и 44, после выполнения программы по генерации оставшихся весов маскирующей части.

Таблица 5

Маскирующая часть весовых коэффициентов нейронов

Номер нейрона	1	2	3	4	5	6	7	8
39	–0,02534	–0,03749	–0,03341	–0,01595	–0,01490	–0,2243	–0,2491	0,1995
44	–0,03749	–0,03341	0,163	–0,2198	–0,2764	0,2538	0,1341	–0,2478
	9	10	11	12	13	14	15	16
39	0,2068	–0,1859	–0,1138	0,177	–0,2377	0,2609	–0,0319	0,3256
44	0,2607	0,1479	–0,0568	–0,2021	–0,0374	0,1489	–0,2796	0,2824

Сумма маскирующей части у обоих нейронов равняется 0, что не ухудшает качества работы нейронной сети биометрической идентификации.

Благодаря дообучению нейронной сети биометрической идентификации с целью маскирования значимых весовых коэффициентов возможно противодействовать атакам Маршалко, так как увеличивается сложность поиска значимых общих связей нейронов из-за наличия маскирующей части нейрона, задача которой запутывать атакующего.

Список литературы

1. Ахметов Б. Б., Иванов А. И., Фунтиков В. А. [и др.]. Технология использования больших нейронных сетей для преобразования нечетких биометрических данных в код ключа доступа. Алматы : LEM, 2014. 144 с.
2. Marshalko G. B. On the security of a neural network-based biometric authentication scheme // Математические вопросы криптографии. 2014. Т. 5. С. 87–98.
3. Крохин И. А. Противодействие атакам Маршалко итерационным дообучением нейронов как способ повышения стойкости биометрической защиты личных криптографических ключей // Безопасность информационных технологий : сб. науч. ст. по материалам II Всерос. науч.-техн. конф. Пенза : Изд-во ПГУ, 2020. С. 61–66.

References

1. Akhmetov B.B., Ivanov A.I., Funtikov V.A. et al. *Tekhnologiya ispol'zovaniya bol'shikh neyronnykh setey dlya preobrazovaniya nechetkikh biometricheskikh dannykh v kod klyucha dostupa = Technology of using large neural networks to convert fuzzy biometric data into an access key code*. Almaty: LEM, 2014:144. (In Russ.)

2. Marshalko G.B. On the security of a neural network-based biometric authentication scheme. *Matematicheskie voprosy kriptografii = Mathematical issues of cryptography*. 2014;5:87–98.
3. Krokhin I.A. Countering Marshalko attacks by iterative training of neurons as a way to increase the durability of biometric protection of personal cryptographic keys. *Bezopasnost' informatsionnykh tekhnologiy: sb. nauch. st. po materialam II Vseros. nauch.-tekhn. konf = Information technology security : collection of scientific articles based on the materials of the II All-Russian scientific and technical conf.* Penza: Izd-vo PGU, 2020:61–66. (In Russ.)

Информация об авторах / Information about the authors

Игорь Алексеевич Крохин
аспирант,
Пензенский государственный технологический университет
(Россия, г. Пенза, проезд Байдукова/ул. Гагарина, 1а/11)
E-mail: kr.ig.sv08@gmail.com

Михаил Юрьевич Михеев
доктор технических наук, профессор,
заведующий кафедрой информационных технологий и систем,
Пензенский государственный технологический университет
(Россия, г. Пенза, проезд Байдукова/ул. Гагарина, 1а/11)
E-mail: mix1959@gmail.com

Igor' A. Krokhin
Postgraduate student,
Penza State Technological University
(1a/11 Baydukov passage/Gagarin street, Penza, Russia)

Mikhail Yu. Mikheev
Doctor of technical sciences, professor,
head of the sub-department of informational technologies and systems,
Penza State Technological University
(1a/11 Baydukov passage/Gagarin street, Penza, Russia)

Авторы заявляют об отсутствии конфликта интересов / The authors declare no conflicts of interests.

Поступила в редакцию/Received 21.12.2021
Поступила после рецензирования/Revised 20.01.2022
Принята к публикации/Accepted 21.02.2022