Application of deep neural networks for security analysis of digital infrastructure components[a]

Alexander Pechenkin[1*], and *Roman* Demidov[1]

[1] Peter the Great St. Petersburg Polytechnic University, Institute of Computer Sciences and Technologies, Department of Computer Systems Information Security, 195251 Polytechnicheskaya st. 29, Russian Federation

Abstract. In this article the authors give a consideration to a problem of detecting errors and vulnerabilities in software components of different digital devices. The article shows an ever-increasing criticality of this problem in the course of time related to development of modern concepts the Industrial Internet and the Industry 4.0. It gives an overview of modern approaches to application of methods of computer-assisted learning and artificial intellect in the sphere of cyber security, problems and prospects of application thereof. A new approach is offered to searching software vulnerabilities on the basis of application of deep learning. The approach is based on building semantically significant vector representations of software code and multistage instructing the deep neural network on revealing hierarchical abstractions in computer code testifying to presence of vulnerabilities. The authors describe specific features of the goal of analyzing software code for presence of vulnerabilities and proceeding thereof it is offered to use a neural network with long short-term memory (LSTM). In order to solve a problem of the learning set, the authors offer to use learning with transfer in case of building vector representations of instructions. The article also provides results of experimental investigations on application of offered solutions.

1 Security of digital infrastructure components

1.1 Trends of developing information and computer sphere

The world industry is now on the verge of the fourth technical revolution [1] denoting a transition to automated digital production controlled by smart systems in real-time mode with permanent interaction with the environment extending beyond the boundaries of one

[a] This work is supported by the Russian Science Foundation under grant No.17-71-10065
[*] Corresponding author: pechenkin@ibks.spbstu.ru

enterprise with a perspective of being associated into a global industrial network of goods and services. The integration of a great number of fields of activity with information and network technologies has brought about a result that a possibility of data exchange and operations with the use of Internet became the most important need of the modern society. The ever-growing number of produced devices in all spheres of vital activities (medical services, power engineering, housing services and utilities, industry, etc.) are digital and get integrated into a unified digital information and computer environment. In this case, the solution of task of providing cyber security is essential for overcoming restrictions in the development and incorporation of up-to-date technologies. For instance, according to Peter Tren, Director General of RSA Security, the provision of cyber security is a top priority for building "smart cities" [2]. Moreover it is relevant in such spheres as unmanned means of transportation, where incorrect electronics operation can bring about catastrophic consequences [3].

The mobile technology and all-round automation have brought about an interesting result – a number of devices connected to Internet a way back has exceeded the number of people inhabiting the Earth. As of the present moment, more than 10 billion devices in the world already use the network protocols for connecting to the Internet, and it means, that the security thereof depends directly on timely detection of network vulnerabilities (vulnerabilities in implementations of network protocols), therefore, they help an intruder remotely perform an arbitrary code at the device connected to the network (including Internet). Everything surrounding us in the digital world of future will become a potential source of cyber threats and a potential target for cyber attacks. The cyber threats appear to be one of the biggest sources of danger for community, economics and states. Therefore, the provision of cyber security of final products is an important and even critical element of digital designing of new devices. It underlines a necessity and relevancy of new methods of searching vulnerabilities in the software of devices.

1.2 Software components analysis for availability of vulnerabilities

The availability of vulnerabilities in the software components of various devices is one of the most critical problems in the sphere of cyber safety. The "Kaspersky laboratory" points out a trend in its forecast for 2017 [4] to conducting goal-oriented attacks of high complexity, which are based most frequently on implementation of vulnerabilities. The methods of detecting vulnerabilities are divided into two classes: static conducting code analysis without its execution, and dynamic based on checking different parameters in the course of SW operation. The modern methods have a number of drawbacks (exponential complexity, availability of limitations for a code to be analyzed, requirement of code availability in high-level programming languages, etc.), which do not allow to efficiently use them in practice. At present there is no universal method, which helps reveal software vulnerabilities in the software encountered actually over acceptable period of time. In the general case the task of searching vulnerabilities is confined to NP-complete task of SAT-feasibility of Boolean formulae. All heuristic approaches to searching vulnerabilities are aimed at revealing not the vulnerabilities proper, but the assigned features (criteria) thereof in advance, and therefore, they have a strictly limited field of application.

Widespread is the proprietary software, which source code corresponds to an intellectual property of this or the other company, which are, as a rule, not interested in publication or transfer of its developments to the third parties. Such a situation contributes to occurrence of both occasional errors and vulnerabilities, and software bugs intentionally introduced into a code. Besides, the analysis of source software text does not allow revealing some types of errors, which can be related to peculiarities of compilators. In some systems (e.g., firmware of transmitters and sensors, field programmable gate array, etc.) the

software components are initially presented in the form of binary codes. It emphasizes the necessity and relevance of developing new methods of searching vulnerabilities based on the analysis of software code being executed.

All the described factors demonstrate a necessity and relevance of developing new technologies for analyzing components of digital infrastructure for availability of vulnerabilities meeting the following requirements: (1) possibility of analyzing programs represented by binary code; (2) intricacy of developed algorithms shall provide for a possibility of analyzing programs by means of modern computation devices over acceptable period of time. The solution of this problem offered in this article is based on modeling cognitive perception of images for presentation of highly-abstract software characteristics. Such a decision is not based on exponential attack of input data and search of expected features of vulnerabilities, at that, the application of neural networks with deep learning will provide revealing deep trends uniting a vulnerable code. The implementation of the offered approach will help get an algorithm, which will identify the root reasons and characteristics hidden in the code, which bring about vulnerabilities over time polynomial from code size.

1.3 Application of machine learning methods in the sphere of cyber security

The recent years were marked with rampant development of methods of machine learning and cognitive analysis and cognitive analysis. In fact, a breakthrough has taken place in different spheres, e.g., such as identification of images (disclosure of faces in photos, identification automobile license plates) and analysis of texts (revealing focus and subjects, machine translation). Fig. 1 shows statistics of registration of rights for intellectual property in the field of cyber security [5].

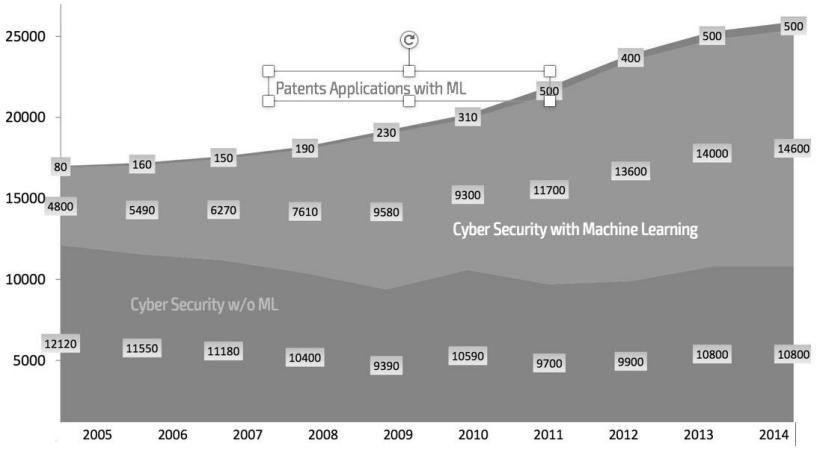

Fig. 1. Statistics of registrations of intellectual property in the sphere of cyber security.

It can be seen that the number of requests for solutions of machine learning remains approximately at the same level, while with the use thereof – it increases sharply in the recent years. In this case a growth of requests for registration of program solutions in the sphere of cyber security based on using methods of machine learning takes place. The methods of machine learning are used in different directions of cyber security: spam filtration, searching bots in social media, revealing botnets, data leak prevention (DLP), analysis of network traffic, analysis of signals from systems of detecting intrusions, etc. [6, 7, 8]. In this case the authors offer an approach to revealing vulnerabilities based on using neural networks and machine learning. Particularly, the development of methods of

machine learning and artificial intellect will help to solve a problem of revealing vulnerabilities in future in great numbers of various digital devices.

2 Neuro-semantic approach to searching vulnerabilities

2.1 Analysis of high-abstract program characteristics

It is proposed to consider vulnerabilities in the code as the abstract code properties as a whole, which get expressed only in terms of presence/absence any machine instructions therein. These properties are based, in turn, on less abstract properties of the same code (Fig. 2). All this shapes a hierarchic structure of the code properties. The purpose consists in developing approach that provides revealing these properties in the incoming code samples with the aim of revealing vulnerabilities.

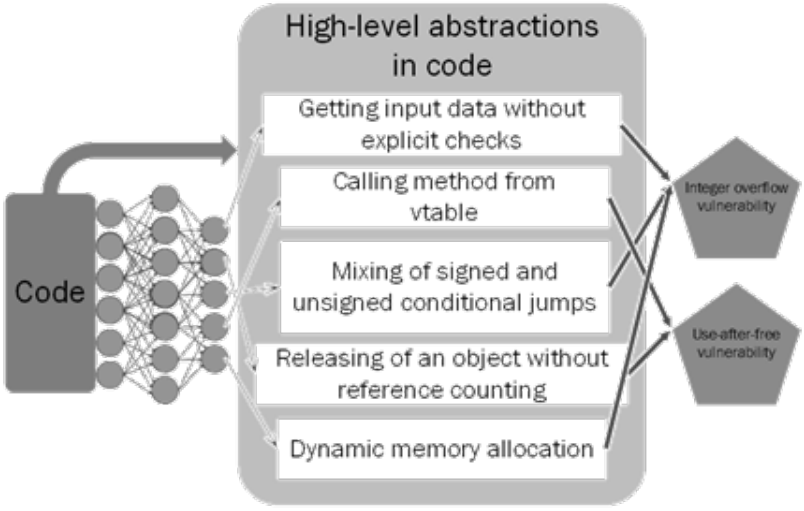

Fig. 2. Analysis of high-abstract program characteristics.

In order to attain this goal, it is recommended to use methods of deep learning of neural networks for building algorithm classifier capable of identifying deep reasons of emergence of vulnerabilities in the machine code and classify the code by classes of vulnerabilities on this basis.

2.2 Building vector representations of program instructions

The application of methods of machine learning for revealing vulnerabilities faces a number of problems; one of such problems is a compilation of learning sample. The availability of a great number of samples is the main requirement for application of machine learning methods. For instance, the methods of machine learning in the sphere of analyzing photos gained momentum particularly after appearance of bases comprising a great number of marked images (indicating what is depicted thereat). Since the majority of companies do not strive after dissemination of information about vulnerabilities, the acquisition of a great number of examples of a vulnerable code is difficult task from organizational point of view. In order to solve this problem, the authors offer to use learning with transferal. It will also help oppose a problem of disappearing gradients in neuronets [8]. Instead of solving a task of searching vulnerabilities with "expensive"

learning data (for which it is difficult to get a big learning sample) initially the auxiliary neural network is being learned on solving simple auxiliary task with "cheap" learning data (for which it is easy to build a great learning set). It is proposed to use emulation of the simplest programs as such a task. A preliminary setting of weights of neural network in the initial layers (Fig. 3, network A+B) takes place as a result of solving an auxiliary task. As a result a representation of individual instructions with coding semantics thereof in the form of dense vectors of real numbers is shaped in the initial layers (Fig. 3, part A) of neural network. Later on the weight coefficients of a part of network with vector representations of instructions (Fig. 3, part A) do not change and get connected to the inlet of neural network for solving the main task of searching vulnerabilities (Fig. 3, network C+D). The stage of building vector representations is described in the article of authors with more details [9].

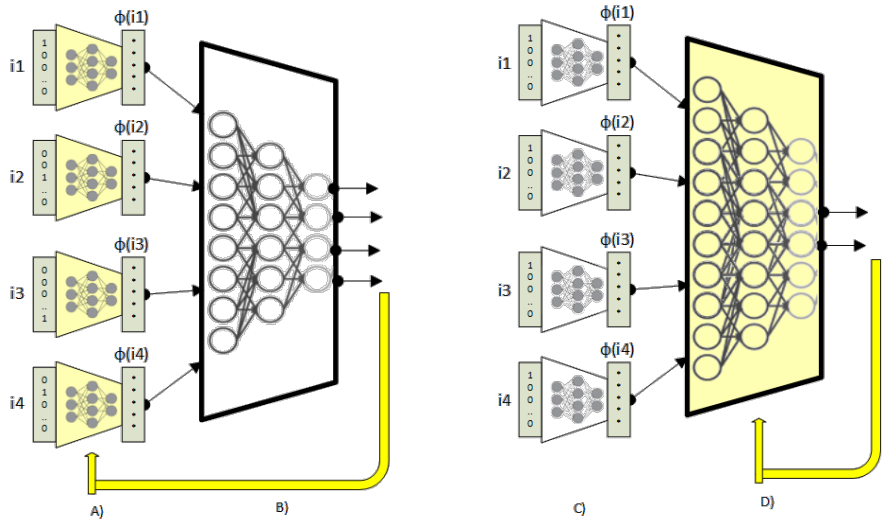

Fig. 3. Left side: stage of building vector representations (A) with the use of network (A+B) for solving auxiliary task. Right side: learning network (C+D) for solving the main task with the use of fixed vector representations (C).

2.3 Architecture of deep neural network

The program code as data form features the key peculiarities:

- discreteness – data on code corresponds to a sequence of locally non-correlated objects;
- availability of semantics of instructions in terms of impact on memory;
- high variability of code – small changes sometime entail radically different program behavior;
- duality of data – presence of instructions, and numerical parameters thereof in the code.

Proceeding from these peculiarities it is offered to use cascade of several layers of recurrent networks LSTM (Long Short-Term Memory) as architecture of neural network for revealing vulnerable code areas [10]. The recurrent neural networks make it possible to process serial data (time rows, series, sequence of non-commutative operations). LSTM networks as a subtype of recurrent networks are tailored for operation with long-term dependencies between the members of incoming sequence the same as it happens in long texts and in code of programs. A fundamental layout of the network is shown in Fig. 4.

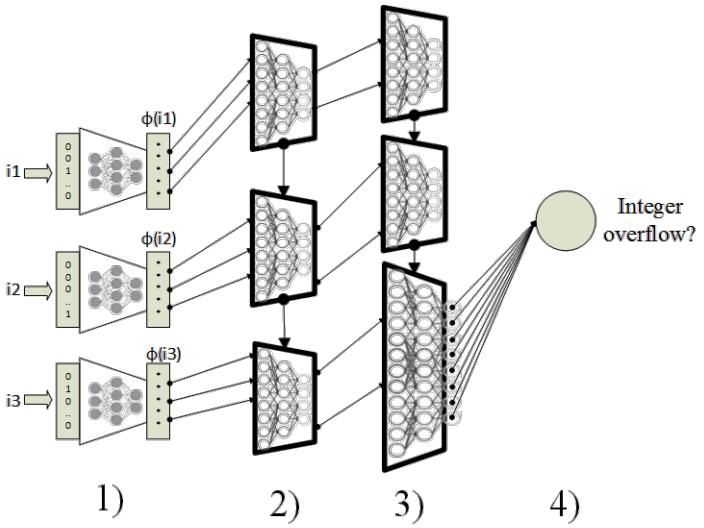

Fig. 4. Architecture of neural network for task of searching vulnerabilities of integer overflow.

A code specimen in the form of sequence of program instructions is fed to the network input. The network consists of the following parts (Fig. 4):

1. Embedding layer. Layers pre-learned earlier, which reflect the indices of instructions and vector representations thereof. The weights of this part of network have been acquired by method of learning with transferal on auxiliary sub-task [11, 12].

2.-3. Layers *consisting of sequence of LSTM cells*. It is a recurrent type of neural networks intended for work with time rows or sequential data in the form of tests. Every LSTM cell in the layer is connected with the next module and receives information of vector representation of current instruction to the input.

4. *Output fully connected layer.* It receives information to the input from the last LSTM module in the last layer and outputs a number from 0 to 1 – probability of presence of vulnerability in the input code sample.

It is possible to classify a new code with respect to availability to vulnerabilities in future by learning such a network in the plurality of marked code samples. The corrective adjustment of weight coefficients is carried out by algorithm of reverse error propagation for the recurrent LSTM networks [13, 14]. A difference between mark for every code example and network response to this example shapes the error signal. In order to minimize the network error, the weights of neural network change in the order reverse to the order of signal passage through the network in the test example. First of all a change takes place in the weights running to the resulting neural of the last layer (in Fig. 4 – 4), then the weights in the last LSTM cell (in Fig. 4 – 3) and ending by the weights in the first such module of the initial layer (in Fig. 4 – 4). Depending on learning parameters, the weights can regenerate only after some test examples, but not after every example. It speeds up learning, however, makes its result less accurate, especially under conditions of deficiency of training examples. The acceleration of learning without losing quality will be attained due to parallelizing process at GPU.

2.4 The data for network machine learning

The performance of approach in general is provided in many respects by data size and integrity for learning at every stage. Different representative sets of training data get shaped

for learning different model parts: vector representations of individual instructions and code classifier as a whole with respect to presence of vulnerabilities therein. The semantics of every instruction is to be derived to the fullest in the process of building vector representations of instructions. In order to do so, the training data shall comprise the exhaustive information about using every instruction under different conditions. The fullest set of marked data is required for correct building of all levels of abstraction of goal-oriented algorithm-classifier of the code. The training code samples comprise a mark on availability/absence of vulnerabilities of definite classes.

The data for learning vector representations correspond to a set of short code sectors together with marks. The learning with transferal is used at this stage, while the marks describe several code properties; the implicit learning of semantics of individual instructions takes place during analysis thereof. The marks depend on the particular auxiliary task and proceed from emulation of code of the corresponding architecture. In general, an indicator of any side effect emerging in the course of its fulfillment is the mark of the code sector. The marks get generated automatically; the learning of semantics of individual instructions takes place at this stage; no presence of code of real programs is required. In order to generate marks to training examples, Open Source projects – Keystone assembler [15], Unicorn emulator are used [16]. The code sectors get shaped randomly with maximization of coverage of possible variants of using machine instructions (Fig. 5).

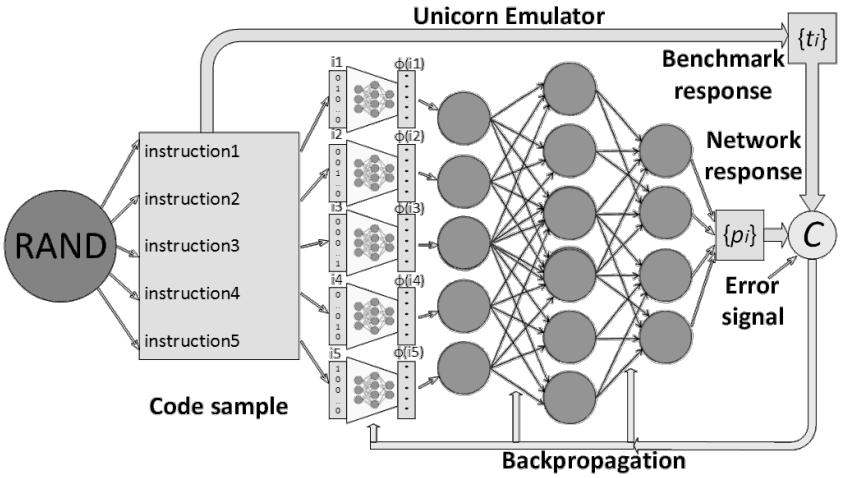

Fig. 5. Learning vector representations using emulation of side effects of code short sectors.

The marked data for learning the main network get shaped from two sources. First of all, the synthetic code examples get shaped automatically with vulnerability and without it. Some classes of vulnerabilities can be entered into a code artificially, equally as vise versa – a vulnerable code can be intentionally corrected with vulnerability removal. Secondly, the data about a vulnerable code will be extracted from data bases on vulnerabilities (CVE, Microsoft Security Bulletin), as well as generated on the basis of renovations (they comprise information about code binary patches). The vulnerable code examples denote a code before patch application, a code without presence of vulnerability is the code after patch application. It helps learn the network to differentiate between "bad" or "good" code and, by the same, identify vulnerabilities in the real code.

3 Experimental results

The authors have carried out experiments demonstrating correct work of the offered approach for searching vulnerabilities of the integer overflow in machine code. The vulnerability of the integer overflow is one of the most frequently encountered and consists in uncontrolled overflow of the integer register as a result of fulfilling arithmetic operations and future use of this register when fulfilling critical operations (e.g., memory allotment). A task being solved by neural network consisted in searching the integer overflow in machine code of different length. A sub-plurality of instructions of processor of x86-archtecture has been used – a limited number of instructions and registers, reduced number of possible numerical values.

Analyzed instructions:

Every instruction is represented by one of five commands:

- nop
- mov REG, REG/NUM
- add REG, REG/NUM
- sub REG, REG/NUM
- call malloc

Three eight-byte registers of general purpose REG = {ax, bx, cx}, all eight-byte numerical values NUM = [0, 1, .., 255] are supported. Collectively there are 2,332 different variants of program instructions.

Input data of neural network:

In this experiment the programs represented by the sequence of instructions are supplied to the input of neural network LSTM (Ref. item 2.3). The last command – a call of function of dynamic memory allotment with the size located in register ax. The initialization of registers prior to code fulfilling has taken place by random values in the range of values [25,.., 32].

Configuration of neural network:

The network architecture in general is described in item 2.3 of this article. The enabling layer compares a real vector of physical dimension 11 with every input instruction. Three LSTM layers, physical dimension of output for LSTM modules: 8, 5, 3, accordingly. The network output in the range [0.1] describes a probability of yes/no decision for the assigned task.

Vector representations of instructions:

The fixed weights in the enabling layer have been acquired in the course of implementing other experiments [9] during learning on the auxiliary task of forecasting the code emulation results. These values are used in this experiment as a carrier of knowledge about instructions as they are.

Learning sample:

The network has been learned for operation with three different program sizes. In order to solve such tasks, three synthetic data sets have been generated (Table 1). Every example has been marked using a back tracing method, the markers have been placed on the presence/absence of vulnerability in the code (taking into account permissible boundaries of external data).

Table 1. Parameters of test sets.

Experiment No.	Length of programs (number of instructions in a program)	Size of learning sample	Size ofchecking sample
1	20	18000	2000
2	30	15000	5000
3	50	6500	500

Learning results:

15 sequential epochs have been used; the network weights have been renovated after every training example in the course of learning. The network has been correctly learned to find vulnerabilities in programs for every used length (Table 2).

Table 2. Parameters of test sets.

Experiment No.	Accuracy at all samples (%)	Accuracy at vulnerable samples only (%)	Accuracy at non-vulnerable samples only (%)
1	79	80	78
2	81	82	80
3	83	84	82

Finding:

The acquired results presented in the table demonstrate the network's capability to classify the incoming code for availability of the integer overflow with a sufficient accuracy. The result has been significantly enhanced by the presence of prebuilt vector representations for instructions under conditions of a small number of examples for learning.

4 Conclusion

The development and implementation of this approach will help develop solutions providing automatic detection of vulnerabilities in the program components of different digital devices over acceptable time without using exponential memory consumption. This is especially relevant under conditions of dashing increase of the number of digital devices accessible through Internet. The incorporation of this approach in practice will help, if necessary, to additionally learn the network with new examples of the vulnerable code (with new types of vulnerabilities). Besides, a modular architecture of computation architecture provides for a possibility of supporting different hardware architectures of processors in the way of learning vector representations for different sets of instructions. These results can be used for advanced detection of and removal of program vulnerabilities in the code of different digital devices being protected.

References

1. K. Schwab, *The Fourth Industrial Revolution* (Crown Business, New York, 2016)
2. P. Tran, Navigating Smart Cities: A Q&A with RSA's Peter Tran [online], Available at: http://www.itproportal.com/features/navigating-smart-cities-a-qa-with-rsas-peter-tran/ (2017)

3. M. Kalinin, P. Zegzhda, D. Zegzhda, Y. Vasiliev, V. Belenko, Proceedings of 7th International Conference on Information and Communication Technology Convergence (ICTC2016) (2016)
4. Kaspersky Security Bulletin 2016, Forecast for 2017: the end "of virus indicators" [online], Available at: https://securelist.ru/files/2016/11/KSB_Predictions_2017_RUS.pdf (2016)
5. L. Rokach, When Cyber Security Meets Machine Learning [online], Available at: http://ucys.ugr.es/jnic2016/docs/MachineLearning_LiorRokachJNIC2016.pdf (2016)
6. CS259D: Data Mining foe Cyber Security [online], Available at: https://web.stanford.edu/class/cs259d/lectures/Session2.pdf (2017)
7. D. Barbara, S. Jajodia, Applications of data mining in computer security [online], Available at: http://www.cs.odu.edu/~mukka/cs795sum10dm/Lecturenotes/Day7/Barbara%20Jajodia%20Data%20Mining%20Book.pdf (2002)
8. R. Pascanu, T. Mikolov, Y. Bengio, Proceedings of the 30th International Conference on International Conference on Machine Learning, on the difficulty of learning recurrent neural networks, **28** (2013)
9. R. Demidov, A. Pechenkin, Approach to software security analysis based on vector representation of a machine code, **4** (2017)
10. H. Sak, A. Senior, F. Beaufays, Proceedings of the Annual Conference of the International Speech Communication Association, Long short-term memory recurrent neural network architectures for large scale acoustic modelling (2014)
11. L. Bottou, Machine Learning, From machine learning to machine reasoning, **94 (2)** (2014)
12. T. Mikolov, W.-T. Yih, G. Zweig, Proceedings of NAACL-HLT “Linguistic regularities in continuous space word representations” (2013)
13. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Proceedings of the IEEE "Gradient-based learning applied to document recognition", **86 (11)** (1998)
14. B. Bakker, IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning, "Reinforcement learning by back propagation through an LSTM model/critic (2007)
15. Keystone, The Ultimate Assembler [online], Available at: http://www.keystone-engine.org/ (2018)
16. Unicorn, The Ultimate CPU emulator [online], Available at: http://www.unicorn-engine.org/ (2018)