An Intrusion Detection System for Network-Initiated Attacks Using a Hybrid Neural Network

Stefanos Koutsoutos, Ioannis T. Christou, Sofoklis Efremidis
Athens Information Technology, 19km Markopoulou Ave. Paiania 19002
Greece
skou@ait.edu.gr, ichr@ait.edu.gr, sefr@ait.edu.gr

Abstract. We present a hybrid system based on a combination of Neural Networks and rule-based matching systems that is capable of detecting network-initiated intrusion attacks on web servers. The system has a strong learning component allowing it to recognize even novel attacks (i.e. attacks it has never seen before) and categorize them as such. The performance of the Neural Network in detecting attacks is very good with success rates of more than 78% in recognizing new attacks. However, because of an alarmingly high false alarm rate that measures more than 90% on normal HTTP traffic carrying image uploads we had to combine the original ANN with a rule-based component that monitors the server's system calls for detecting unusual activity. A final component combines the two systems to make the final decision on whether to raise an intrusion alarm or not. We report on the results we got from our approach and future directions for this research.

1 Introduction

Intrusion Detection is a major issue that every administrator has to deal with effectively so as to maintain proper operation of their Internet-connected servers. Network security and Intrusion Detection was studied as early as 40 years ago [6], but with the current growth of the Internet and the number of attackers, it has taken on a very prominent role in the fields of computer and communications security ([7, 12, 18, 21]). For this reason, the problem has been widely studied from a number of different perspectives. Data Mining approaches have been used to detect unusual activities on the servers ([8,10]). System calls are monitored in [9] to discover patterns that are characteristic of an intrusion. Actual Intrusions (when another person actually takes over in an unauthorized way someone else's workstation) have

Please use the following format when citing this chapter:
Koutsoutos, Stefanos, Christou, Ioannis, Efremidis, Sofoklis, 2006, in IFIP International Federation for Information Processing, Volume 204, Artificial Intelligence Applications and Innovations, eds. Maglogiannis, I., Karpouzis, K., Bramer, M., (Boston: Springer), pp. 228–235

also been studied in [16,17,20] and dealt with a number of different approaches, many times utilizing Artificial Intelligence techniques.

The idea of using Artifical Neural Networks or ANNs ([3, 22]) for finding novel attacks is not new ([1, 2, 4, 13, 14]). The usual approach is to train a neural network with the behavior of the system and let it recognize any behavior which substantially differs from the one considered as normal. The behavior is described to a neural network with a number of features, like system calls invocation, rates of system call invocation, etc. In this paper *we explore the possibility of using a hybrid system based on neural networks which will be trained and accept data directly from the data network and try to identify attacks using this data, in real time, by continually monitoring the network as well as the server state.*

The ultimate goal is to be able to identify both known and unseen attacks, on the fly, mainly using their binary signature, as this travels through the network. There are numerous kinds of attacks, each of which has a completely different structure of signature. Because different applications may use different protocols, network traffic may substantially differ from one case to another. This variety renders the task of building a neural network, which will be able to respond to every possible attack, essentially infeasible. In order to overcome this kind of problem, *we chose to fix the target application/protocol used as well as the kind of attacks we are targeting. The application chosen is an Apache web server (Linux/i386), and the kind of attacks we are targeting is remote code injection attacks, exploiting stack or heap overflows, etc*

2 Detecting Attacks in the Network Traffic of a Web Server

One of the first issues in the design of a Neural Network is the type of inputs it will accept, or the features to be used in the representation. Supplying raw network data to the ANN is not a good idea, because of their size, but even more importantly because it is then really hard for the neural network to learn classifying patterns in this kind of data. Finally the ANN input's layer size must be large enough to be able to accept an amount of data that will enable it to judge correctly whether an attack is included in the network packet or not. Having an "aggregator function" that does not essentially loose any important features in our network data for classification purposes (operating on the packets right before they enter the ANN) may both minimize their size and keep their "structure" intact. The aggregator function we used was:

$$f(m) = \frac{\sum_{i=0}^{w} m_i \cdot m_i}{w \cdot 255^2}$$

In the above formula, the parameter m_i refers to the i'th byte of the incoming packet.. The window, 'w', is a system parameter, which specifies the amount of bytes "grouped" together in one number. The denominator normalizes the output in the interval [0,1]. This function's main goal is to minimize the data load passed on to the

ANN. The feature extractor's output for the same data, but different window values keeps on the same levels, although as the window grows, the output graph tends to attenuate

2.1 Training the ANN

A component which listens to any inbound web server traffic, will face the HTTP protocol, and mainly page requests (HTTP GET requests), but some requests may well be file upload requests or other form-handling requests (HTTP POST requests). These are also included in the traffic that can be seen. Using the feature extractor introduced above, we can verify that pure text (ASCII) won't ever cause the ANN to give an output larger that 0.22. This of course means that the task of differentiating between text and binary data is an easy task. The HTTP protocol is a text based protocol, but of course not all binary traffic is malicious. File uploading, (such as the uploading of multimedia material) supported by many sites, may result in the ANN seeing images or other kinds of binary files. So, in order to differentiate between this category and malicious code, we trained an ANN with jpeg image file parts for one class and actual code for the other. The library which we used for taking the code exemplars was libc because it's the one closer to the system that any other, so it's more probable to look like code to be injected in an application.
In the general case of measuring the ability of the neural network to distinguish between the two classes, we found out that the ANN can indeed classify code as such, but it's difficult for it to correctly identify image binary data from potentially harmful code. In our tests, we trained several neural networks, using standard BackPropagation [22]. The input layer sizes selected were 10 and 20. The hidden layer sizes ranged from half the input layer size to its double. The epochs each ANN had gone through were 1000, 5000 and 10000. In total we trained 30 ANNs with 800 exemplars per group. Figure 1 shows the performance graph of all those ANNs. The line marked 't/set-1' refers to the train-set data which should be classified as malicious traffic, while 't/set-0' refers to normal traffic. The line 'class-1' refers to the NN performance on real malicious traffic, which has not been seen by the NN during its training. Accordingly, 'class-0' is normal traffic, which the NN has not seen during its training.

The correct classification of real malicious traffic (class-1) doesn't exceed 80%, for any of the NNs trained. This means that 80% of completely novel attacks are correctly identified as such. We can also see that the percentage of correct classification of real normal traffic (class-0) is always below that of class-1, for all the case where the performance for class-1 is above 50%.

By augmenting a few the training set with more code (t/set-1) and less image data (t/set-0), we got an interesting outcome, for the same NN architectures. The train set had 300 exemplars of (artificial) normal traffic (t/set-0) and 500 ones of code (t/set-1). The results are shown in Figure 2.

What is important to notice in these two graphs, is that although the correct classification percentage for malicious traffic (class-1) has improved substantially, the one for normal traffic remained at the same levels. Nevertheless, although the ANN can correctly classify potentially malicious code as such, it is also often

mislead when presented with jpeg images, giving false alarms at an unacceptably high rate.

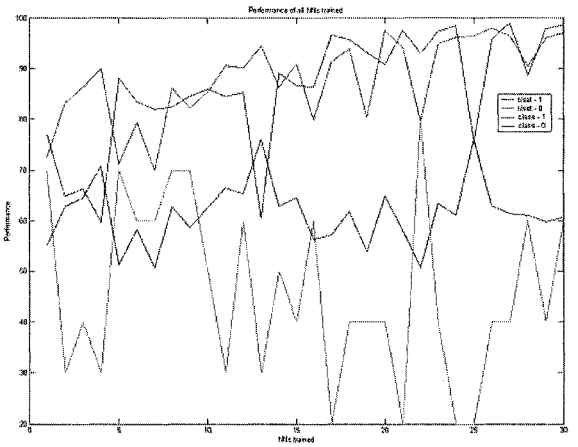

Fig. 1. ANN Performance using 800 exemplars per category

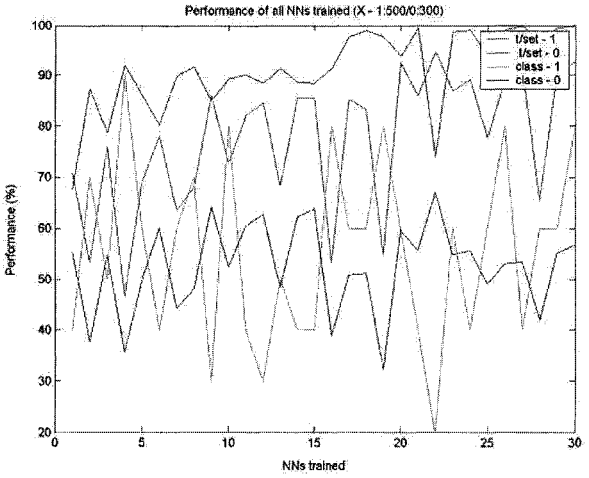

Fig. 2. ANN Performance using 500 and 300 exemplars per category

3 Architecture of a Hybrid Intrusion Detection System

As mentioned above, using the trained ANN alone as an IDS would suffer serious performance problems with normal traffic. Essentially, because of the very high false

alarms rates, the system would be useless as any system administrator would quickly decide to ignore all alarms raised by the ANN, including of course the real ones. Nevertheless, the system has a lot of advantages, including its minimal size, which is translated to less load for the system it runs on, and its ability to detect new attacks. It can also associate an attack with its source host. In order to put these advantages into good use, we decided to create a hybrid system which combines this network classifier with some other component. That second component accepts input from a different source and helps identify real attacks on the server. Then, a third component coordinates the actions of the previous two sub-systems, and taking their outputs into account, makes the final decision about whether to issue an alarm or not.

For this purpose, we built a tool that monitors the system calls invoked by the Apache web server that is our target server application. Monitoring system call invocations for unusual patterns has been studied before for detecting intrusion attempts with reasonable success [9]. The system is rule-based. Figure 3 shows a schema of our architecture.

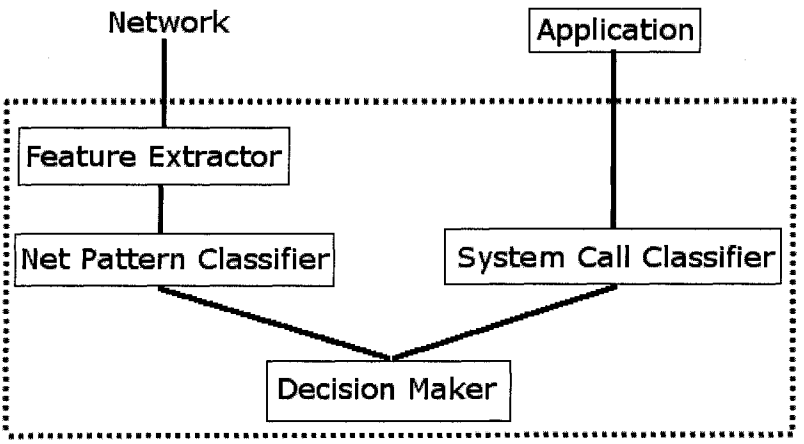

Fig. 3. Architecture of the Hybrid IDS for Network-initiated Intrusion Attacks

3.1 Implementation & Performance of a Rule-based System Call Classifier

In order to successfully create and test a Rule-based system-call classifier, we needed a training set containing some patterns of misbehavior as well as the usual patterns of normal system call invocations under normal circumstances. For this reason, we created a web application as a CGI program in C with two exploitable buffer overflows and installed it on our Apache test server running under Linux. Afterwards, a number of attacks, together with normal requests were created. In total each test set contained 100 requests carrying no attack and no file upload, 100

requests carrying an attack but no file upload, 100 with file upload but no attack, and 100 requests carrying an attack with a file upload

The Decision Maker component works by continuously listening to the output of the network classifier, and when this output exceeds a certain threshold (set empirically to 0.8), the Decision Maker asks the opinion of the System-Call Classifier. If the output of the second component also exceeds a certain threshold, then an alarm is raised. We monitored the Apache system calls under normal conditions (no attacks), and produced a series of "acceptable" system calls stored in a configuration file. In the first version of this Rule-based subsystem, the system call classifier monitors the server and all its children processes for any invocation that is not described in the configuration file. If a call not matching any of the ones in the configuration file is detected the subsystem raises an alarm. The output of this rule-based classifier is always binary in nature (giving always a 0 or a 1). The results for the overall system are shown in table 1. In the first column of the table, NF/NA means „requests containing No File uploads and No Attack", NF/WA means „request containing No File uploads but containing an Attack", „WF/NA" means „request containing File upload, No Attack", and finally „WF/WA" means „request containing File upload, With Attack". The columns heading „Min" and „Max" indicate what was the minimum or maximum number of alarms raised per request in each test-set, and „Total" describes the total number of alarms raised for the whole test-set. Each test-set, as mentioned previously, comprised of 100 requests.

Table 1. Performance of the Hybrid IDS using a Rule-based system call classifier.

Reqsts:	**NetClassifier Alarms:**			**Decision Maker:**		
	Total	Min.	Max	Total	Min	Max
NF/NA	0	0	0	0	0	0
NF/WA	73	1	3	65	1	2
WF/NA	98	2	5	0	0	0
WF/WA	97	1	5	58	1	2

As can be seen from the above table, the false alarms rate of the overall system is essentially reduced to zero. The IDS is now capable of detecting approximately 60% of new -never before seen- attacks. Another benefit of this approach is that it is independent of the nature of new attacks. The system administrator only needs to update the rule-based subsystem's configuration file when a new web application or web service is installed on the server, by gathering some data on what constitutes normal system call invocations with the new application, an operation that can be very easily automated.

So, the system performance while being essentially usable, still leaves room for improvement. In the next and final section we describe the directions we are currently taking to improve upon the current system implementation.

4 Future Directions

Because it was noted that often the false alarms raised by the network classifier had a lower than many of the real attacks output level, an easy modification to the Decision Maker is to assume the following strategy: when the network classifier's output is above a certain very high threshold TH, always raise an alarm without even consulting the system call classifier; when the network classifier's output is above a certain threshold TL, but below TH, then consult the system call classifier and combine their results.

Yet another line of research we are actively pursuing is that of detecting „long-lasting" attacks by better monitoring system-calls; in particular, having a larger time window to work with, a rule-based induction system such as ID3 and derivatives, or SLIPPER2 [23] can be used to classify *system call sequences* which should likely give good indications on whether an attack is in progress.

Finally, we are in the process of investigating the effectiveness of alternatives to BackPropagation as the learning-from-errors mechanism such as the GeneRec algorithm, that also happens to have a more plausible biological basis [11], even though it seems to require more computing power.

References

1. W.Lee, S.Stolfo, K.Mok: "A Data Mining Framework for Building Intrusion Detection Models" - Proceedings of the 1999 IEEE Symposium on Security and Privacy (May 1999)
2. H.Debar, M. Becker, D. Siboni: "A Neural Network Component for an Intrusion Detection System" - Proceedings of the 1998 National Information Systems Security Conference (NISSC'98) October 5-8 1998. Arlington, VA.
3. LiMin Fu: "A Neural Network Model for Learning Rule-Based Systems" - Proceedings of the 1992 International Joint Conference on Neural Networks, I-343:I- 348. [R, L].
4. H. Teng, K. Chen, S. Lu: "Adaptive Real-time Anomaly Detection Using Inductively Generated Sequential Patterns" - Proceedings of the IEEE Symposium on Research in Security and Privacy, pages 278-284, Oakland CA, May 1990.
5. T. Lane, C. Brodley: "Approaches to Online Learning and Concept Drift for User Identification in Computer Security" - Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, pp. 259-263 (1998).
6. J.Anderson: "Computer Security Threat Monitoring and Surveillance" - Tech. Rep., James P Anderson Co., Fort Washington, PA, Apr. 1980.
7. A. Lazarevic, P. Dokas, L. Ertoz, V. Kumar, J. Srivastava, P. Tan: "Cyber Threat Analysis – A Key Enabling Technology for the Objective Force (A case study in Network Intrusion Detection)" - Proceedings 23rd Army Science Conference, Orlando, FL, December 2002.
8. P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, P. Tan: "Data Mining for Network Intrusion Detection" - Tutorial at the Pacific-Asia Conference on Knowledge Discovery in Databases, Seoul, April 30, 2003.

9. C. Warrender, S. Forrest, B. Pearlmutter: "Detecting Intrusions using System Calls – Alternative Data Models" - IEEE Symposium on Security and Privacy (1998).
10. L. Ertoz, E. Eilertson, A. Lazarevic, P. Tan, P. Dokas, V. Kumar, J. Srivastava: "Detection and Summarization of Novel Network Attacks Using Data Mining" – Minnesota INtrusion Detection System (MINDS) Technical Report, 2003.
11. R. C. O'Reilly, and Y. Munakata: "Computational Explorations in Cognitive Neuroscience: Understanding the Mind by Simulating the Brain", MIT Press, Boston, MA, 2000.
12. T. Lunt: "Real-Time Intrusion Detection" - Technical report, Computer Science Laboratory, SRI international, Menlo Park, CA, February 1992.
13. J. Frank: "Artificial Intelligence and Intrusion Detection – Current and Future Directions" - Technical Report, Division of Comp. Science, University of California at Davis, 1994.
14. J. Ryan, M. Lin, R. Miikkulainen: "Intrusion Detection with Neural Networks" - AI Approaches to Fraud Detection and Risk Management: Papers from the 1997 AAAI Workshop (Providence, Rhode Island), pp. 72-79. Menlo Park, CA: AAAI.
15. M. Mahoney, P. Chan: "Learning Rules for Anomaly Detection of Hostile Network Traffic" - Proceedings of the Third IEEE International Conference on Data Mining, p.601, November 19-22, 2003
16. A. Ghosh, A. Schwartzbard, M. Schatz: "Learning Program Behavior Profiles for Intrusion Detection" - Reliable Software Technologies Corporation, 1999.
17. J. Cannady: "Artificial Neural Networks for Misuse Detection" - National Information Systems Security Conference (1998).
18. L. Ertoz, A. Lazarevic, E. Eilertson, P. Tan, P. Dokas, V. Kumar, J. Srivastava: "Protecting Against Cyber Threats in Networked Information Systems" - SPIE Annual Symposium on AeroSense, Battlespace Digitization and Network Centric Systems III, Orlando, FL (2003)
19. L. Lankewicz, M. Benard: "Real-time Anomaly Detection Using a Nonparametric Pattern Recognition Approach" - Proceedings of the of 7th Computer Security Applications conf., San Antonio, TX, 1991.
20. J. Shavlik, M. Shavlik: "Selection, Combination and Evaluation of Effective Software Sensors for Detecting Abnormal Usage on Computers Running Windows NT/2000" – Shavlik Technologies Apr. 2002.
21. K. Ilgun: "USTAT: A Real-time Intrusion Detection System for UNIX" Proceedings of the IEEE Symposium on Security and Privacy,Oak-land, CA, May 1993.
22. N. Bose, P. Liang: "Neural Network Fundamentals with Graphs, Algorithms, and Applications" – McGraw-Hill, 1996.
23. W. W. Cohen, and Y. Singer, "Simple Fast & Effective Rule Learner", AAAI/IAAI 1999, pp. 335-3