

INTCTD: A Peer-to-Peer Approach for Intrusion Detection

Catalin L. Dumitrescu

Computer Science Department, The University of Chicago

Email: catalind@cs.uchicago.edu

Abstract

In this paper we propose a peer-to-peer (P2P) prototype (INTCTD) for intrusion detection over an overlay network. INTCTD is a distributed system based on neural networks for detecting network traffic anomalies and for modifying dynamically the network resource access policies. Automated learning and online knowledge sharing are employed among the participating nodes, while the distillation of the network traffic is performed by each individual each node. This approach for local analysis of the network traffic gives the opportunity for utilization of simple automated learners at each node and the reduction of the amount of information exchanged among the peers.

1 Introduction

Intrusion detection is an important problem in the security arena where external attacks on overlay networks are on the rise. The necessity of intrusion detection is triggered by the increasing dependence of people on various overlay networks that exist over the Internet. The wide spread usage of overlay network in Internet also increases the possibility of attackers to gain illicit access to computers all over the world. In order to protect such communities from security hazards, better intrusion detection mechanisms have been devised. We consider here that once a security system is able to detect an attack, it could also take automatically corrective measures without the necessity of a human administrator to update access policies for each individual resource.

This work targets mainly the possibility of detecting remote attacks against overlay networks either from an external source or from within the system. The approach we adopt is to process the raw data of the network traffic and detect any suspicious behavior. In addition, once a new access policy is learned, it is also exchanged with all the other participants in the overlay network for improving the global security of the system. We also note that considering a single point of enforcing various access policies for an overlay network might be difficult in practice. Usually an overlay

network is composed of resources spread over large geographic areas and placed under different administrative domains. Enforcing a common access policy over all these resources in a centralized manner is practically impossible – not all domains will have the same rules for their internal networks and the common access policies have to be enforced at the resource level.

2 Background Information

A real time and perfect intrusion detection for overlay networks has become a more and more complex task, which cannot be done anymore off-line or manually by a system administrator. The creativity of attackers correlated with the range of computer hardware and software components, the lack of a mechanism for automatically verification of software correction, and the fast ever-changing nature of computing world have increased the needs for effectively identifying intrusions.

2.1 Intrusion Detection Classifications

The existence of geographically distributed overlay networks increases the complexity problem of intrusion detection. Various approaches to intrusion detection have been proposed and applied over the years. Statistical approaches, Predictive Pattern Generation, Expert Systems, State Transition Analysis, Pattern Matching and Autonomous Agents are some of them. Two of the main criteria in classifying the efforts done in the intrusion detection area over time are:

- **Data Source Based Classification** deals with the data collection methods for filtering. *Host Based Systems* use data from a single host for detecting intrusions. Such systems were in place from the early days of computers, and they are represented by simple file logs used to trace how users are using the system. *Multi-host Based Systems* collect data from multiple hosts for detecting intrusions. This approach uses the same strategy as the one described above, somehow collecting data and avoiding situations when intruders can remove local traces of their presence. *Network Based*

Systems enhance the capabilities of multi-host Based Systems one step further by taking network traffic data into account along with audit data from one or more hosts, in order to detect intrusions.

- **Intrusion Detection Based Classification** takes into account the way collected information is used and compared with information already known. *Anomaly Detection Systems* detect intrusions by scanning for activity that is different from a user's or system's normal behavior. This approach defines the normal functioning of a system. Any differences from the normal patterns will trigger off the suspect. *Misuse Detection Systems* detect intrusions by looking for activity that corresponds to known intrusion techniques (signatures) or system vulnerabilities. When it is possible to define the pattern of all possible attacks, misuse detection is a good alternative.

2.2 NeuNets Based Intrusion Detection

In this paper we consider a NeuNet for traffic analysis. More exactly, several NeuNets are trained over some period of time and learn the expected network traffic pattern for an overlay network. Afterwards, the NeuNets must detect any anomalies in the traffic pattern and, by means of the layered infrastructure, perform adequate actions (generate new access policies or trigger actions that have to be revised by a human operator). The advantage of NeuNets consists in their flexibility and capacity to accommodate noisy data (as network traffic is bound to be). The disadvantage of using NeuNets is their incapacity to provide the way a decision was reached. Practically, a NeuNet may discover and report an anomaly but it has no means of conveying the actual trigger that sets off the action. Another disadvantage is the training data. Since the entire learning depends on the training data, large amounts of sane, correct data have to be fed to the NeuNet before it can be considered ready.

3 INTCTD Design

INTCTD is build on a layered architecture that filters the network traffic in several steps. The top layers are represented by higher level interfaces, that offer independent access methods to specific lower level modules and third parties components. The resource access policies are handled by the higher layers of the architecture. One of the problems is the integration and interfacing with generic NeuNets. Thus, our solution was to support *mediator-like pattern* interfaces that also expose characteristics specific to the *facade pattern*. The mediator is encapsulating a set of interfaces for various module interactions and it is also promoting loose coupling by keeping objects from referring directly each other [3].

3.1 INTCTD Modules

Now we turn our attention to the basic building blocks of our system. Each high level layer has a well defined interface, which is made available to both neighbor layers. We believe that separating the concepts at the architectural level enhances INTCTD's portability and also its flexibility for further extensions while having minimal impact on the implementation of how modules are exchanging information inside the system.

Collection Modules: The collection modules are responsible for gathering data from the host operating systems and for broadcasting meaningful knowledge to their peers. INTCTD implements the interfacing with the host operating system by means of either a special system log files or kernel-tunned modules. The first approach offers the advantage of testing the system also off-line ([2]) and of verifying the accuracy of the learners. In the second approach, traffic logs are directly collected by registering a kernel-tunned module and collecting data at the firewall level. In the second approach, a kernel space submodule gathers the network traffic, and a user space submodule incorporates into INTCTD. The kernel space submodule is registered as a new listener for incoming/outgoing traffic, buffers the traffic in a compact format and provides it trough the */proc* virtual file system.

Learner Modules: The learner modules are the most computing consummative parts of the system, and INTCTD is practically built around them. Our solution was to use NeuNets, but any other approach can be plugged in (fuzzy logic or statistical approaches [10]). The difference of the higher level layer is in how it uses the lower level information. While for all the other modules there is a matching between the request and the service provided, the learner modules are considered homogeneous. Thus, they are able to handle only generic data. In our implementation any request is sent to all registered learners and responses are composed in an single response by using a priority-based algorithm.

Statement-Generator Modules: The pro-active part of INTCTD is the statement-generator module. It is supposed to generate security-adds in real time and to be inserted in the firewall chains of a system or to be considered by human administrators. For the specific scenario of the Linux firewall, this module is a rule generator of the type: *accept / reject / deny*. For any operating system without fire-walling support, INTCTD must provide the mechanisms to drop unwanted network traffic. As an additional feature, we mention the possibility of storing analyzed traffic for later check-up by using a module that logs all meaningful information.

Communication Modules: The current communication is based on a TCP level module. We consider that this ap-

proach is somehow costly regarding the performances in a real time system. A UDP low level module seems to be more appropriate for such an application. The TCP proposed solution is not only a little awkward about handling connections (one for each data packet exchanged), but it also introduces high overhead in network traffic. In our previous work, we concluded that the rate of transfer advantages clearly the UDP based solution, while the cross rate might represent a problem without any communication recovery mechanism.

Encryption Modules: The last part of INTCTD presented here is the encryption module. This is required for data exchanges over untrusted network links. Encryption becomes available as soon as a module is registered within INTCTD. Without such a module, the entire communication is insecure and may be intercepted or even tampered during transition between the overlay points. We have considered so far a public/private key-based implementation.

4 Validation Studies

In order to validate our hypothesis and the viability of the INTCTD framework, we have performed several experiments starting from already public collected traces. These traces were obtained from the *MIT Lincoln Laboratory* [2]. Several network traces and audit data are provided representing a simulated network with six hosts, a cisco router and 2 hubs. The traces considered for validating INTCTD consisted of 5 days of network traffic corresponding to a business week from 8:00 to 16:00. The following day traces were used for INTCTD testing. The total number of packets taken in account was around 200,000 for one week of network traffic (data taken in consideration was generated in 1998). The training iterations of the NeuNet required several hours to complete. In the studied case we forced INTCTD's learners to reconsider a trace at most 1000 times.

4.1 Experimental Results

As mentioned, from the considered network traces, only seven event records were selected for training and testing. These seven characteristics were selected because they are the easiest to capture from any nowadays kernel. They also provide a sufficiently accurate description about the type of ongoing communication. The seven characteristics are: *Time of the day*, *Duration*, *Source Port*, *Destination Port*, *Source Address*, *Protocol Type*, and *Data Length*. While this minimal testing scenario, based only on the network traffic characteristics, is not a complete intrusion detection checking, the results demonstrate clearly the potential of a NeuNet approach for anomaly detection in network traffic.

Tuning a NeuNet for a specific task is yet one of the hardest problem. In figure 1 is presented the speed of learning

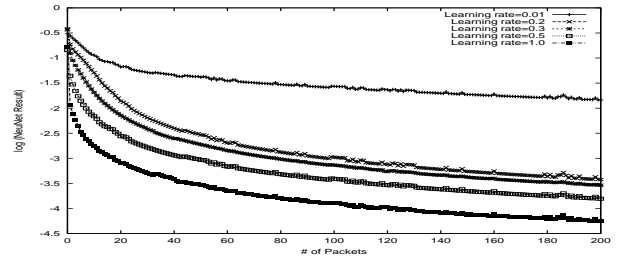


Figure 1. Learning Landscape

for different learning rate parameters. The smaller the rate is, the slower the network learns. In the same time, a higher learning rate generates instability during the learning process and in this case the entire mechanism is unable to fulfill its job. In our case, the data fed to the NeuNet was not highly distributed, making the NeuNet sometimes unable to learn fine grained information when its complexity was low (number of nodes, number of hidden layers). Unfortunately, a large NeuNet requires a higher amount of computation resources for its back-propagation algorithm, in order to update all weights. As can be easily observed, a large NeuNet learns *better* the patterns, but it requires also longer time intervals, which is not feasible.

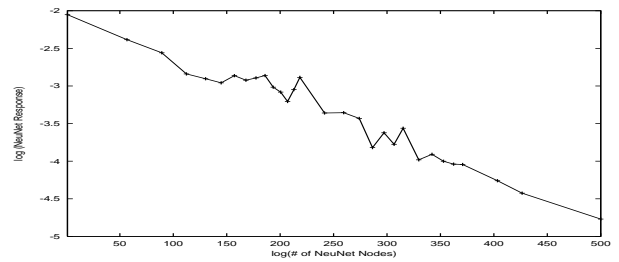


Figure 2. Learning Performance

The above conclusions conducted us to the decision of implementing INTCTD as a distributed intrusion detection system. Practically, a collection of small NeuNets deployed on each system require fewer computation power. Also, the amount of network traffic that has to be analyzed is a few orders of magnitude smaller. Another advantage is that each detector has theoretically full access to local resources, making possible to add at any time additional auditing modules that outperform a centralized system without direct access to the rest of the network.

To prove a NeuNet is able to detect anomalies in a system during run-time, we trained the network with a typical multi-day traffic, and let the NeuNet learn everything as a correct pattern. As already stated, we did not have any control on this learning stages beyond the provided input information and controlling the shape of internal structure of the NeuNet. Thus, a drawback of the NeuNet approach is that once deployed in a specified context, the system incorporates specific knowledge which cannot be used in other

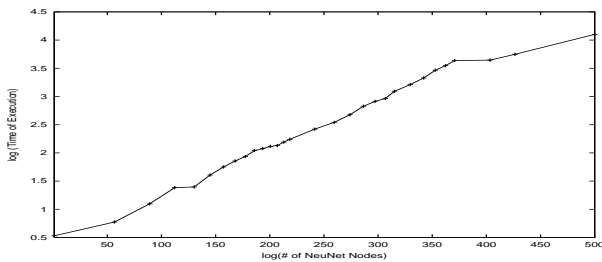


Figure 3. Required Training Time

contexts the same way. We consider this problem one of the main reasons for the avoidance of using NeuNets in intrusion detection compared for example with the speech recognition domain, where once a NeuNet is trained it will perform always better wherever it is used later.

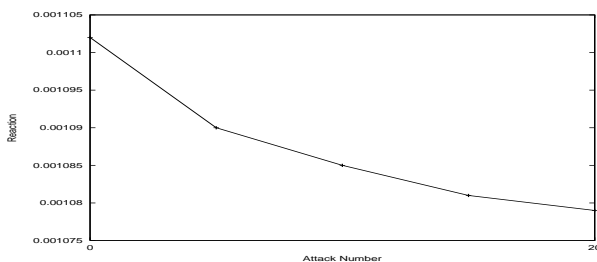


Figure 4. Anomaly Detection Behavior

Because the network has no mechanism of direct reward, the reinforcement has to be provided in a different way. The solution proposed here is the use of a set-point to control when things are forgotten and reminded. In our case, the solution is straightforward by modifying the learning error rate of the NeuNet system. When the network is “living” good moments, without traffic, the error rate is slightly increasing (reminding rules), while when the network is learning by analyzing the traffic, the error rate is slightly decreased.

5 Conclusions

There has been a great amount of work done on intrusion detection in general, but not much work which relates NeuNets to intrusion detection. This is probably because a certain result achieved by a NeuNet cannot be traced and explained. Statistical methods can do so and thus are at an advantage. Some papers describe the use of NeuNets in anomaly detection ([4], [5]), but they do not analyze network data to identify an intruder. They are mainly concerned with program behavior profiles for intrusion detection. One project that actually implemented a NeuNet for intrusion detection for a distributed system ([8]) concentrates in fact mostly on misuse detection. The paper mainly aims to show that NeuNets are a viable method for intrusion detection. Their system is designed to capture raw data

off the network and to use some of the fields like source, destination etc. to convert it into a form that can be fed to the NeuNet. Our work differs from this paper in terms of the philosophy of the whole project in trying to design a decentralized system where many simple automated learning agents coordinate among themselves.

Acknowledgment

The author would like to thank Kavitha Ranganathan for the contribution to this work.

References

- [1] *Bro Software*, LBL Network Research Group Papers. <http://www-nrg.ee.lbl.gov/nrg-papers.html>
- [2] *DARPA Intrusion Detection Evaluation* - <http://ideval.ll.mit.edu>
- [3] Gamma E., Helm R., Jonshon R., Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software* - Addison-Wesley Publishing Company Inc., 1994
- [4] Ghosh A.K., Wanken J., and Charron J. *Detecting anomalous and unknown intrusions against programs* - In Proceedings of the 1998 Annual Computer Security Applications Conference (ACSAC '98), December 1998.
- [5] Ghosh A.K., Schwartzbard A., and Schatz M. *Learning Program Behavior Profiles for Intrusion Detection* - In USENIX Workshop on Intrusion Detection and Network Monitoring. USENIX Association.
- [6] Kumar S., Spanfford E. *A Pattern Matching Model for Misuse Intrusion Detection* - In proceedings of the 17th National Computer Security Conference, pages 11-21.
- [7] Vern, P. *Bro: A System for Detecting Network Intruders in Real-Time* - LBL Berkeley, January 14, 1998
- [8] Cannady, J. *Artificial Neural Networks for Misuse Detection* - <http://citeseer.nj.nec.com/cannady98artificial.html>
- [9] Storck J., Hochreiter S., and Schmidhuber J., . *Reinforcement-driven information acquisition in non-deterministic environments*. In Proc. ICANN'95, vol. 2, pages 159-164. EC2 & CIE, Paris, 1995.
- [10] Zuev D., and Moore A., *Traffic Classification using a Statistical Approach*, in the Proceedings of Sixth Passive and Active Measurement Workshop (PAM 2005), March/April 2005, Boston, MA