A Synthetic Fraud Data Generation Methodology

Emilie Lundin, Håkan Kvarnström[⋆], and Erland Jonsson

Department of Computer Engineering
Chalmers University of Technology
412 96 Göteborg, Sweden
{emilie,erland.jonsson}@ce.chalmers.se
hakan.k.kvarnstrom@telia.se

Abstract. In many cases synthetic data is more suitable than authentic data for the testing and training of fraud detection systems. At the same time synthetic data suffers from some drawbacks originating from the fact that it is indeed synthetic and may not have the realism of authentic data. In order to counter this disadvantage, we have developed a method for generating synthetic data that is derived from authentic data. We identify the important characteristics of authentic data and the frauds we want to detect and generate synthetic data with these properties.

Keywords: fraud detection, synthetic test data, data generation methodology, user simulation, system simulation

1 Introduction

Fraud detection is the process of automated analysis of data from a service with the goal of revealing attempts to use the service without paying or in some other way illicitly benefit from the service. When designing a fraud detection system, it is essential to have suitable data for evaluation and testing. This data must be representative of normal and attack behavior in the target system since detection systems can be very sensitive to variations in input data.

Using synthetic data for evaluation, training and testing gives several advantages compared to using authentic data. Data properties of synthetic data can be tailored to meet various conditions not available in authentic data sets. On the other hand purely synthetic data suffers from the fact that they may be quite unrealistic and will consequently not properly reflect the properties of authentic data. Thus, the aim of this work has been to develop a methodology for generating synthetic data with realistic properties and this is achieved by using authentic data as a property "seed". Thus, the method uses statistical properties from smaller amounts of authentic data and generates large amounts of synthetic data preserving parameters important for fraud detection and the training of fraud detection systems. Examples of such parameters are user and service behavior.

[⋆] The author is also with Telia Research AB, SE-123 86 Farsta, Sweden

The rest of this paper is organized as follows. Related work is summarized and discussed in section 2. Advantages and motivation for using synthetic data can be found in section 3 and our method for generation of synthetic data is presented in section 4. Conclusions can be found in section 5.

2 Related Work

There are very few papers which focus on properties of test data. Most papers describing training and test data focus on testing a specific detection prototype, or is about benchmarking and comparing detection systems. Often, data is only mentioned briefly. In most papers, (manipulated) authentic data is used.

The absence of research on synthetic data for fraud detection applications has led us to study work in the intrusion detection area. In [10] we discuss several similarities between fraud and intrusion detection systems. However, there are some differences that need to be considered. Intrusion detection is performed on log data from operating systems, applications, and networks to detect malicious actions in a computer system. Fraud detection is normally performed on log data from a specific service to find people trying to gain unauthorized benefits from the service. Fraud detection is often more specialized and adapted to a service as opposed to intrusion detection which is more general. Roughly, fraud detection can be seen as application-specific intrusion detection. We believe that the way test and training data is retrieved and used is similar and the same methods for generating data can be used.

In the following two sections, we review some papers that present test methods using authentic and synthetic test data. We describe how the data is retrieved, used, and manipulated, and how it compares to our method.

2.1 Authentic Data

In the JAM project [2], some papers have been written that actually focus on test and training data properties. For example, [4], and [15] describe experiments on varying the fraud rate in training data and also introduces a cost model where each transaction is associated with a cost. In [15] which is an unpublished technical report, the data and the manipulation of data is described more thoroughly, but this information is shortened in the published papers based on the same experiments. For the experiments two large sets (500000 records each) of authentic credit card transactions are used. The transactions are labeled as fraudulent or legitimate and the fraud rate is around 20%. A cooperation with two banks provided the data. The labeling seems to have been performed by bank employees and the authors mention that some of the unclear cases were indeed labeled as fraudulent. The authors concluded that the optimal rate of fraudulent vs. normal events in training data depends on the cost model and the learning algorithm used. For their applications, the desired rate is shown to be close to 50:50. The fraud categories presented in their work are likely to be representative of future fraud. However, the process for labeling data is not described.

In [3], data is described but not analyzed in much detail. The data used for the experiments in this paper is GSM Toll Tickets from Vodafone. It consists of calls made by 300 normal users during two months. This data is "sanitized", but it is not specified how this is done. It also consists of calls made by 300 fraudulent users but it is not explained how these fraudulent users are selected. The paper describe what information the Toll Tickets contain although the ratio between normal and fraudulent events is probably not realistic.

2.2 Synthetic Data

Probably the biggest effort in testing and comparing intrusion detection systems is made by DARPA in 1998 and 1999 [11]. A great deal of work has been spent on generating large amounts of test and training data for this project. The generation of data is best described in [8]. The test data contains network traffic and system call log files (SUN BSM log files) from a simulated large computer network. Both attacks and background data has been generated synthetically, but the background data is said to be similar to sampling data from a number of Air Force bases. Background data is generated mainly by using software automata simulating the usage of different services. Attack data is generated by running attack scripts. The database of attacks used is described in [9]. Some more complicated background data and attacks are injected live into the system. No extensive analysis of the quality of the generated data has been made. Some efforts in automating the data generation further is described in [7].

McHugh [13] criticizes the lack of validation of test data in the 1998 DARPA evaluation. The process of generating data is vaguely described, and it is difficult to determine the quality of the data. There are no guarantees that the attacks used are really representative and realistic. Also, he questions whether the background data is really representative of the user behavior in the computer systems used and if the behavior in these systems represent user behavior in other computer systems. Another question he raises is whether attacks are realistically distributed in the background data. Some improvements were made in the 1999 experiments but many of these issues are still not fully addressed.

Debar et al. [6] developed a generic intrusion detection testbed. They show how to simulate user behavior with a finite state automata to obtain normal traffic, but state that this is only a practical approach if the set of user commands is limited. Instead, they use recorded live data from user sessions, which has been replayed inside the workbench. They also describe how they create attack scripts to use in the testing process.

Maxion and Tan [12] discuss the effects of more or less irregular background data. They generate "random" background data with different degrees of regularity and show that it affects the false alarm rate drastically. The paper also measures regularity in real-world log data.

Chung et al. [5] claim that concurrent and distributed intrusions are likely to occur in a computer system, and therefore, should be included in test data for intrusion detection systems. They have developed a tool that parallelizes attack scripts to simulate concurrent intrusions.

Puketza et al. [14] describe a software platform for testing intrusion detection systems. They have used the UNIX package *expect* to generate synthetic user sessions. However, they do not analyze the quality of this data to any great extent.

From the discussion above we conclude that most projects seem to use synthetic data due to lack of authentic data or lack of data having desired properties. In the DARPA project [1], it was necessary to generate synthetic data since it was not possible to retrieve the desired amount of data. Also, most projects seem to suffer from a shortage of real attacks, which makes it necessary to inject fraud/intrusions synthetically.

3 Using Synthetic Data for Evaluation

Training and testing of fraud detection systems require data which has certain properties. Access to authentic data from real services is often preferred although it may suffer from lack of control of what fraud cases it contains and the amount of data may not be sufficient. Another possibility is to use synthetic data.

Synthetic data can be defined as data that is generated by simulated users in a simulated system, performing simulated actions. This definition can be relaxed to include humans performing simulated actions on a system. Simulated actions means that people (or a program) perform actions according to a specification created by the experiment organizers, and not act according to their normal behavior. This specification should reflect the desired behavior of the system. In this paper, we use the wider definition of synthetic data. It depends on the situation whether it is better to simulate user behavior using a software automata or to hire people to generate background data and attacks. There is also a choice between using a fully simulated system, a real system or a mix of real and simulated system components.

3.1 Rationale

Use of authentic data is not always a viable solution for evaluation of detection systems. For future services (e.g. services that are planned or under development) authentic data may not exist or may only be available in small quantities. In these situations, synthetic data is the only possible solution for conducting tests. Moreover, expected fraud cases for services under development are not present in the authentic data as the service has no real users. Under these circumstances, synthetic data containing expected fraudulent user behavior must be generated for testing of the detection system.

An advantage of synthetic data is that it can be designed to demonstrate properties, or include attacks, not available in the authentic data. This gives a high degree of freedom during testing and training. In addition, synthetic data may be generated to cover extensive periods of time which could have taken months or years to collect from the real target system. Also, a large number of users can be efficiently simulated, even though the real system only has a few of them. This makes scalability tests possible.

3.2 Data for Training Fraud Detection Systems

The process of detecting fraud involves analyzing large amounts of data, looking for signs of fraudulent behavior. Intelligent techniques such as neural networks and other AI (Artificial Intelligence) techniques can be used to decide whether an event or a group of events indicate fraudulent behavior. Common to most intelligent techniques is that extensive training needs to be performed. During training, data similar to that of the target system must be available. In addition, the data must be labeled (i.e. all events must be categorized as normal or fraudulent) so that the detection algorithm can learn to distinguish normal usage from fraudulent. We have identified a number of properties that characterize good training data.

- Data is labeled, i.e. we have exact knowledge of which attacks are included in the data.
- The attacks in the input data represent the attacks we expect to occur in the target system. (Not necessarily the same attacks that currently occur in the system.)
- The number and distribution of attacks in the background data (fraud/normal data ratio) are adapted to the detection mechanism. Some detection methods perform better if they are trained on data where attacks are over-represented. In [4], it is shown that varying the amount of attacks in the data will greatly affect the training process of the detection algorithms.
- The amount of data is large enough. In particular, certain AI algorithms need huge amounts of training data.

These properties indicate that synthetic data can be a better choice for training fraud detection systems.

3.3 Data for Testing Fraud Detection Systems

The detection algorithms need to be tested on data sets containing expected fraud in the system. Testing the detection capability of the algorithm may require a different data set than used during training, even though most of the properties for training data is valid also for test data. For example, it is easier to test the system if data is labeled. Some additional important properties for test data are stated here.

- The number and distribution of attacks in the background data (fraud/normal data ratio) should be realistic.
- The attacks in the input data are realistically integrated in the background data. For example, time stamp of an attack, time between attacks, and time between parts of an attack may affect detection results.
- Normal (background) data should have similar statistical properties as authentic data from the target system. Different behavior in the system may affect detection performance drastically. In [12], it is shown that the false alarm rate rises sharply when background data becomes more irregular. This would indicate that test data is often system specific.

Testing a fraud detection system involves many diverse activities. Scalability tests need to be performed to see whether the systems can handle current and future data volumes. Again, large authentic data sets may not be available to realistically conduct such tests. Various stress tests can be conducted to test the system's ability to withstand sudden changes in data volume or the characteristics of the data. For example, the system can be tested to determine its susceptibility to parameter changes. These changes could be triggered by sudden changes of the environment in which the detection system operates or by faults and errors in the collected data. By using synthetic data, faults and other unexpected events can be injected to study the susceptibility of the detection system and its ability operate under harsh conditions.

4 Data Generation Methodology

There is a great deal of complexity in synthetic data generation and it is time-consuming to create the necessary components. Therefore, a methodology is needed to structure the work and to point out the choices that have to be made.

The main components needed to automate the data generation process are specifications of desired user behavior in the system, a user/attacker simulator, and a system simulator. The goal of the methodology is to guide the production of these components. The starting-point of the methodology is the collection of information about the anticipated behavior in the target system. The methodology includes both background and attack data generation. Therefore, we need both information about possible attacks as well as normal usage. This data serves as basis for user and system modeling.

4.1 Methodology Overview

In figure 1, the methodology for generating the data generation components is pictured. The first step is the *collection of data* that should be representative of the anticipated behavior in the target system. The data may consist of authentic background data, background data from similar systems, authentic attacks, and other collections of possible attacks. The second step is to *analyze the collected data* and identify important properties such as user classes, statistics of usage, attack characteristics, and statistics of system behavior. In step 3, the information from the previous step is used to identify parameters that need to be preserved to be able to detect the anticipated attacks, and to *create user and attacker profiles* that conforms to the parameter statistics. A *user model* is created in step 4. This model must be sophisticated enough to preserve the selected profile parameters. Also, the attacks are modeled in this step. The user and attacks simulators implements the models. In step 5, the system is modeled. The model must be accurate enough to produce log data of the same type as the target system for the input user actions. The system simulator is implemented according to this model. It is important that it is possible to configure the user and system models using variable input parameters in order to change the properties of the generated data during operation.

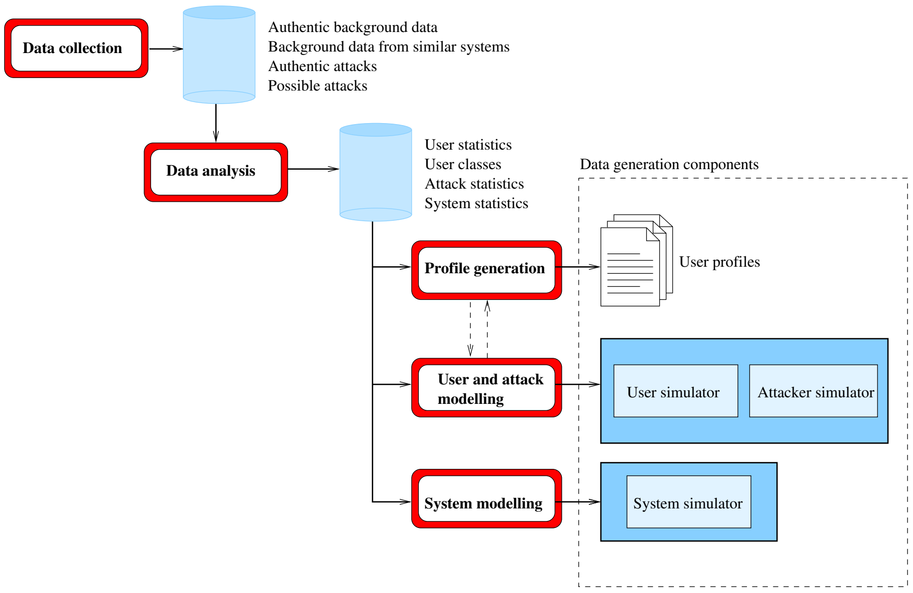

Fig. 1. Synthetic log data generation method

In the following sections, each of the steps in the data methodology are discussed and methods for implementing them are suggested.

The division into steps with well defined interfaces reduces the complexity of the task and makes it possible for different groups of people to work on different tasks. It is possible to run some of the work in parallel. For example, as soon as some authentic background data is collected and analyzed, it is possible to start working on both the user model and the system model. In parallel to the user and system modeling, attack data may be collected and analyzed to create the attack model.

It is possible to use people instead of a user simulator to create the user actions. It is also possible to use the whole, or parts of the real system instead of a system simulator. In some situations, this may be preferable, especially if the system or user behavior is very complex, and needs to be modeled in great detail. However, there are some disadvantages in using people and real hardware and software in the generation process.

If only simulation programs are used, it is possible to make the data generation process fully automatic. In the DARPA evaluation [8], they use humans to inject attacks, and a great deal of system hardware and software. They admit that their data generation procedure required much manual administration and attention during the simulation runs.

An advantage of using fully simulated components, is that it has the possibility to become very scalable, both concerning the number of users and the simulation time period. If humans are used to generate background data, each person can only perform the tasks of one or a few simulated users. If we want to use humans to inject attacks, the simulation time can not be much faster than real-time. Also, if real software and hardware is used, these components may limit the simulation speed considerably.

The rest of this section presents and discusses the different steps of the methodology in some detail and describes the final creation of the synthetic log data.

4.2 Data Collection

The data we need as a starting point in the data generation method are samples of background data and attacks representative of the anticipated behavior on the target system. The output log data from the target system is the input to the detection system. This is the type of data that should be produced in the synthetic data generation process. Therefore, it is convenient to have samples available of authentic data from the target system. Hopefully, this data also contains information about user and system behavior which is representative. This may be the most valuable source of data for the generation. If too small amounts of authentic data is available, it is also possible to collect data from similar services. In each case, we need to determine if this data is applicable to our service. Even if we have authentic data, these may not be representative, e.g. because the number of users are going to increase or functions in the system are changed before the detection system will be in operation.

Authentic attacks are often not available. Therefore, we need to collect them in other ways. In the intrusion detection area, databases of known attacks are more or less publicly available. In the fraud area, they are often more service specific, and we may need to "invent" possible attack scenarios or adapt known frauds from other types of services to our situation. Collected attacks may be injected into the target system to get corresponding log data. It is important that the log data can be labeled to know exactly which entries corresponds to a specific attack.

It is difficult to know that properties of the collected data corresponds to the environment that the detection system will operate in. On the other hand, this is not a problem specific for synthetic data. If the detection system is trained and tested on data with the "wrong" properties, the detection will not be correct irrespective of the type of input data. We must predict the future behavior of users and hope that the prediction is close enough to reality.

The output from this step may have many different formats. Some of it may be labeled log data on the same format as will be used as input to the FDS. It may also be databases of attacks with only some information about each attack.

4.3 Data Analysis

The next step is to analyze the collected data. The goal of this work is to get a picture of how the system is used and how user actions and attacks show up in log data.

One task is to identify classes of users with similar behavior. Exploratory Data Analysis (EDA) [16] is an existing set of ideas on how to study data sets to uncover the underlying structure, find important variables, detect anomalies etc. We believe that techniques based on those ideas could be used for this purpose. This may include use of visualization tools and/or clustering methods. It is important to use several classes of user behavior to get diversity in the generated data.

Another task is to identify important attack features, i.e. parameters that are useful for detection of the anticipated attacks. For example, to detect an attacker guessing passwords on a system, the number of failed logins within a certain time interval is a useful parameter. Also, system statistics must be examined. For example, response time and amount of network traffic generated by different events may be useful for the system modeling.

The log data generated by the target system should be examined to determine if it is adequate for effective detection. If it is not adequate, it may be necessary to implement additional logging mechanisms and collect new data.

The output from this step is user classes and a number of statistics for different aspects of user, attacker, and system characteristics.

4.4 Profile Generation

The next step is to identify important parameters for user behavior in the statistics. One way to identify these parameters is to study the features needed to detect expected frauds. These features must have correct statistical properties in the generated data to be useful for detection. One example of a fraud indicator would be users logging in at night that usually only log in during daytime. In this case, we must preserve the statistical distribution of logins during the day for the simulated users. Also, correlation between parameters may be fraud indicators and must be preserved.

When we know what parameters we have and what properties we want to preserve, collected data is used to find statistical values for them. There are more or less automatic tools available for calculating values for parameters and fitting data to statistical distributions.

Output from this step is files for different user classes containing values for all parameters that are required for the user simulation. A cooperation between this step and the user modeling step is needed to adapt the format of the profiles to fit the user model. For example, there are different ways to model the frequency of a specific event. Either, it can be defined as the expected number of events per time interval, or it can be defined as the expected time until the next event.

4.5 Modeling the User

The user and attacker behavior can be very complex to model. To limit the complexity, we should bear in mind that it is only of interest to simulate actions that will affect log data. It is in general easier to model the behavior of users in a service which is less complex. Behavior profiles are appropriate to use as configuration data to the user simulator. This way, we can easily adapt the simulator to changes in behavior and add new user classes.

There are several ways to model user behavior. One way is to use a finite state machine. This is suggested by Debar et al. [6] and also used in the DARPA evaluation [11]. This is a suitable method if user behavior is not too complex or if only a limited number of parameters need to be preserved in the background data. Also, it is possible to generate great amounts of data for many simulated users. However, it may be very time consuming to model users and attacks if the behavior needs to be modeled in great detail.

Another way is to "record" real user sessions and attack sessions and replay them in the system simulator. This method is suggested by Puketza et al. [14]. If the behavior is complex, this may be a better method than modeling the users with a state machine. The problems are that the system simulator must be advanced enough to handle this type of input and it may be difficult to generate large enough amounts of user sessions.

A third way is to use test suites developed for testing functionality in operating systems. This is suggested by Debar et al. [6]. The test suites are not representative of real user behavior but generate all sorts of events that may be very infrequent in a real-life system. The advantage of this method is that it does not require much work to implement, and it may be useful for evaluating the false alarm rate when behavior is very irregular. However, it is questionable if this type of data is useful for testing fraud detection systems.

The output of this step is "lists" of user actions in a format suitable to use as input to the system simulator.

4.6 Modeling the System

The simulated system must be able to produce output data similar to that of the target system, where the similarity can be restricted to those features that are necessary for the fraud detection. The simplest way to model the service realistically would be to set up a replica of all system components. This would give us perfectly accurate log data as long as the "users" are behaving realistically. However, it may require a great deal of resources if we want to simulate a large system, and there are other disadvantages as mentioned before.

A more realistic way to model the system would be to make a hardware replica of the central system components, but implement the clients in software. One single machine can simulate numerous clients. This is the method used by DARPA [11]. There are two major problems with this solution. Firstly, it is not trivial to simulate traffic from many different IP addresses from one machine, but it can be done. Secondly, it is not possible to speed up the simulation time as the client software must send out the traffic in real-time if the rest of the components are to respond realistically.

A third solution would be to implement a model of the service in software. Some of the real service software can be used, but we can experience problems if we want to run the simulation at non real-time speed. The advantages of this solution are that we have full control of the simulation process and can speed up the simulation as fast as the computing platform allows. Furthermore, simulation processes can be quickly and easily implemented for low complexity services. The disadvantage is that it may take some time and effort to study the behavior of the service to be able to implement the required functionality. There is also a risk of oversimplification resulting in significant differences between synthetic and authentic data.

The output of this step is data on the same format as the data produced by the target system.

4.7 Creation of Synthetic Log Data

The basic data generation methodology was presented in figure 1. Figure 2 below shows in more detail how the data generation components interact in the generation process.

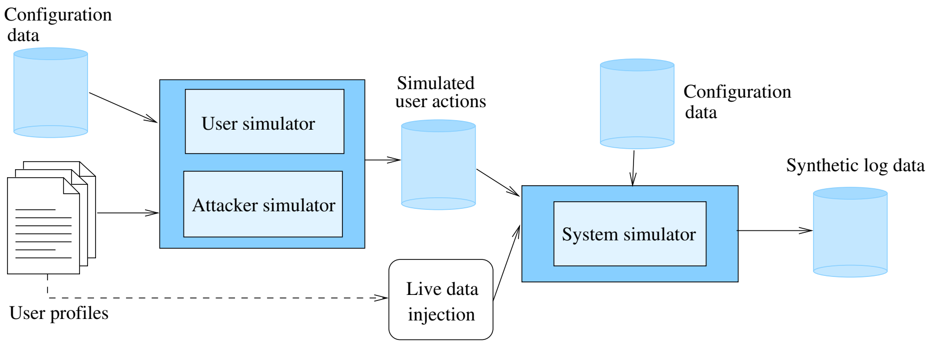

Fig. 2. The synthetic log data generation process

The *user profiles* are used as input to the *user and attacker simulator*. The *user and attacker simulator* generates user actions that are fed to the *system simulator*. The *system simulator* creates the final synthetic test data and the *configuration data for the user and attacker simulator* contains information that controls the generation of normal user and attack actions. For example, it may be the start time and stop time for simulation, the number of normal users from different user classes, the number of attacks of each type, and the start time and stop time for different attacks. The *configuration data for the system simulator* contains for example information about the number of clients that should be simulated, statistics about reply times and traffic amounts for different events, and behavior in the presence of certain attack events. Parameters that we want to vary between different simulation runs should be included in the configuration data. This means that we can generate batches of synthetic data for different purposes without reprogramming the simulators.

It is possible to complement the user simulation with *live data injection*, e.g. to generate more complex series of actions or to generate deviating behavior for special stress tests.

5 Conclusions

The results presented in this paper form a foundation for generation of synthetic test data based on authentic data for fraud detection systems. By starting out from small sets of authentic log data, appropriate synthetic log data can be created in large amounts. Properties of parameters in the initial data are preserved or can be tailored to meet the needs of testing and training of fraud detection systems. In the near future we aim to use our method in practice with authentic log data collected from a pilot-test video-on-demand application.

References

1. DARPA Intrusion Detection Evaluation. http://www.ll.mit.edu/IST/ideval/ The main web page for the DARPA evaluation experiments. July 2001.
2. JAM project homepage. http://www.cs.columbia.edu/ sal/JAM/PROJECT/ July 2001.
3. Peter Burge, John Shawe-Taylor, Yves Moreau, Bart Preneel, Christof Stoermann, and Chris Cooke. Fraud Detection and Management in Mobile Telecommunications Networks. In *Proceedings of the European Conference on Security and Detection ECOS 97*, pages 91–96, London, April 28-30 1997
4. Philip K. Chan, Wei Fan, Andreas L. Prodromidis, and Salvatore J. Stolfo. Distributed Data Mining in Credit Card Fraud Detection. *IEEE Intelligent Systems*, 14(6), Nov/Dec 1999.
5. Mandy Chung, Nicholas J. Puketza, Ronald A. Olsson, Biswanath Mukherjee. Simulating Concurrent Intrusions for Testing Intrusion Detection Systems: Parallelizing Intrusions. In *Proceedings of the 1995 National Information Systems Security Conference*, pages 173-183. Baltimore, Maryland, October 10-13 1995.
6. H. Debar, M. Dacier, A. Wespi, and S. Lampart. An Experimentation Workbench for Intrusion Detection Systems. Technical Report RZ2998, IBM Research Division, Zurich Research Laboratory, Zurich, Switzerland, March 1998.
7. Joshua Haines, Lee Rossey, Rich Lippmann, and Robert Cunnigham. Extending the 1999 Evaluation. In *Proceedings of DISCEX 2001*, Anaheim, CA, June 11-12 2001.
8. Joshua W. Haines, Richard P. Lippmann, David J. Fried, Eushiuan Tran, Steve Boswell, and Marc A. Zissman. 1999 DARPA Intrusion Detection System Evaluation: Design and Procedures. Technical Report 1062, MIT Lincoln Laboratory, February 2001.
9. Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Master's thesis, MIT, 1999.
10. Håkan Kvarnström, Emilie Lundin, and Erland Jonsson. Combining fraud and intrusion detection - meeting new requirements. In *Proceedings of the fifth Nordic Workshop on Secure IT systems (NordSec2000)*, Reykjavik, Iceland, October 12-13 2000.

11. Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579-595, October 2000. Elsevier Science B.V.
12. Roy A. Maxion, and Kymie M.C. Tan. Benchmarking Anomaly-Based Detection Systems. In International Conference on Dependable Systems and Networks, pages 623-630, New York, New York, June 2000. IEEE Computer Society Press.
13. John McHugh. The 1998 Lincoln Laboratory IDS Evaluation: A Critique. In *Recent Advances in Intrusion Detection, Third International Workshop, RAID 2000*, pages 145-161, Toulouse, France, October 2-4 2000. Lecture Notes in Computer Science #1907, Springer-Verlag, Berlin.
14. Nicholas J. Puketza, Kui Zhang, Mandy Chung, Biswanath Mukherjee, and Ronald A. Olsson. A Methodology for Testing Intrusion Detection Systems. *Software Engineering*, 22(10):719-729, 1996.
15. Salvatore Stolfo, Wei Fan, Andreas Prodromidis, Wenke Lee, Shelly Tselepis, and Philip K. Chan. Agent-based Fraud and Intrusion Detection in Financial Systems. Technical report, 1998. Available at:
http://www.cs.columbia.edu/ wfan/research.html.
16. John W. Tukey. *Exploratory Data Analysis.* Addison Wesley College, 1997.