Synthesizing Test Data for Fraud Detection Systems

Emilie Lundin Barse Håkan Kvarnström* Erland Jonsson
Department of Computer Engineering
Chalmers University of Technology
412 96 Göteborg, Sweden
{*emilie,hkv,erland.jonsson*}@*ce.chalmers.se*

Abstract

This paper reports an experiment aimed at generating synthetic test data for fraud detection in an IP based video-on-demand service. The data generation verifies a methodology previously developed by the present authors [7] that ensures that important statistical properties of the authentic data are preserved by using authentic normal data and fraud as a seed for generating synthetic data. This enables us to create realistic behavior profiles for users and attackers. The data can also be used to train the fraud detection system itself, thus creating the necessary adaptation of the system to a specific environment. Here we aim to verify the usability and applicability of the synthetic data, by using them to train a fraud detection system. The system is then exposed to a set of authentic data to measure parameters such as detection capability and false alarm rate as well as to a corresponding set of synthetic data, and the results are compared.

1. Introduction

Fraud detection is becoming increasingly important in revealing and limiting revenue loss due to fraud. Fraudsters aim to use services without paying or illicitly benefit from the service in other ways, causing service providers financial damage. To reduce losses due to fraud, one can deploy a fraud detection system. However, without tuning and thorough testing, the detection system may cost more in terms of human investigation of all the false alarms than the gain from reduction of fraud. Test data suitable for evaluating detection schemes, mechanisms and systems are essential to meet these requirements. The data must be representative of normal and attack behavior in the target system since detection systems can, and should, be very sensitive to variations in input data.

*Håkan Kvarnström is also with TeliaSonera AB, SE-123 86 Farsta, Sweden

Using synthetic data for evaluation, training and testing offers several advantages over using authentic data. Properties of synthetic data can be tailored to meet various conditions not available in authentic data sets. There are at least three application areas for synthetic data. The first is to train and adapt a fraud detection system (FDS) to a specific environment. Some FDSs require large amounts of data for training, including large amounts of fraud examples, which are normally not available in the authentic data from the service. The second application area is to test the properties of a FDS by injecting variations of known frauds or new frauds into synthetic data to study how this affects performance parameters, such as the detection rate. The false alarm rate may also be tested by varying background data, where background data is defined as normal usage with no attacks. The third application area is to compare FDSs in a benchmarking situation.

The aim of this work is to test the feasibility of generating and using synthetic data for training and testing a fraud detection system. Our synthetic data generation is based on the method proposed in [7], where we use small amounts of authentic log data to generate a large amount of synthetic data. The method identifies important statistical properties and aims to preserve parameters important for training and detection, such as user and service behavior. We apply the data generation method on an IP based video-on-demand (VoD) service running in a pilot test environment with real customers. The authentic data collected from the service are used to generate synthetic log files containing both normal and fraudulent user behavior. The properties of the synthetic log files are verified by visualizing them and using them to train and test a fraud detection prototype.

Synthetic data are not commonly used in the fraud detection area, although, the use of manipulated authentic data is discussed in some papers, e.g. [2] and [1]. In the intrusion detection area, more work has been done using synthetic test data. ([5] discusses similarities between fraud and intrusion detection systems.) Huge amounts of synthetic test data were generated in the 1998 and 1999 DARPA intrusion detection evaluations [4]. Debar et al. [3] developed a generic intrusion detection testbed, and suggest the use of a finite state automata to simulate user behavior. While this methodology is rather close to our approach, they did not use this method in their testbed. They declared it practical only if the set of user commands is limited. Instead, they used recorded live data from user sessions. Puketza et al. [10] describe a software platform for testing intrusion detection systems where they simulate user sessions using the UNIX package *expect*. However, none of these methods give sufficient control of the data properties of the synthetic data. We believe that our method can provide better data properties that are needed for training and testing in many situations. Furthermore, our method provides scalability of log data in both amount of users and time period.

Some interesting work has been done on measuring characteristics in data and how they affect the detection systems, e.g. Lee and Xiang [6], and Tan and Maxion [11]. In [8], Maxion and Tan generate "random" synthetic data with different degrees of regularity and show that it affects the false alarm rate drastically. The methods proposed in these papers may be useful for synthetic data validation, but have not been used in this paper.

Below, Section 2 and 3 give a a summary of the method proposed in [7]. The rest of the paper describes how we apply the method on the VoD service and the verification of the generated synthetic data.

2. Motivation for using synthetic data

This section gives the benefits of using synthetic data for various types of testing and explains why authentic data are not a solution in some cases.

2.1. Why not use authentic data?

Authentic data cannot be used in some cases for a number of reasons. The target service may still be under development and thus produce irregular or only small amounts of authentic data. We also have no control over what fraud cases the data contain. Furthermore, it may be impossible or at least very difficult to acquire the amount of or type of data needed for tests. This in turn may be due to the fact that only a limited number of users are available or that we do not know whether the data set contains any frauds.

2.2. Benefits of synthetic data

Synthetic data can be defined as data that are generated by simulated users in a simulated system, performing simulated actions. The simulation may involve human actions to some extent or be an entirely automated process.

Synthetic data can be designed to demonstrate certain key properties or to include attacks not available in the authentic data, giving a high degree of freedom during testing and training. Synthetic data can cover extensive periods of time or represent large numbers of users, a necessary property to train some of the more "intelligent" detection schemes.

In [7], we provide an elaborate discussion of data properties that are important for training and testing fraud detection systems. These are summarized in the following bullets:

- Data need to be labeled, i.e. we need to have exact knowledge of the attacks included in the data.
- The attacks found in the input data are representative of the attacks we expect to find 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) must be adapted to the detection mechanism. Some detection methods perform better if they are trained with data in which attacks are overrepresented.
- The amount of data must be of a sufficient size. In particular, certain AI algorithms need huge amounts of training data to perform well.
- For testing, the number and distribution of attacks in the background data (fraud/normal data ratio) should be realistic.
- For testing, it is important that attacks in the input data are realistically integrated in the background data. For example, the time stamp of an attack, time between attacks and the 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 drastically affect detection performance.

3. Data generation methodology

Synthetic data were generated by the methodology described in [7]. For completeness, we briefly introduce the method and refer to the original work for details.

The main components necessary for automating 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 is the collection of information about the anticipated user behavior in the target system. The methodology includes generating both background and attack data and thus it is necessary to have information about possible attacks and normal usage. These data serve as the basis for user and system modeling.

Figure 1 illustrates the methodology. The first step is the *collection of data* that should be representative of the anticipated behavior of the target system. Data may consist of authentic background data from the target system, 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 must be preserved to be able to detect the anticipated attacks and to *create user and attacker profiles* that conform to the parameter statistics. A *user model* is created in step 4. This must be sophisticated enough to preserve the selected profile parameters. Attackers are also modeled in this step. The user and attacker simulators implement the models. The system is modeled in step 5, and this model must be accurate enough to produce equivalent log data as the target system for the same type of input user actions. The system simulator is then implemented according to this model.

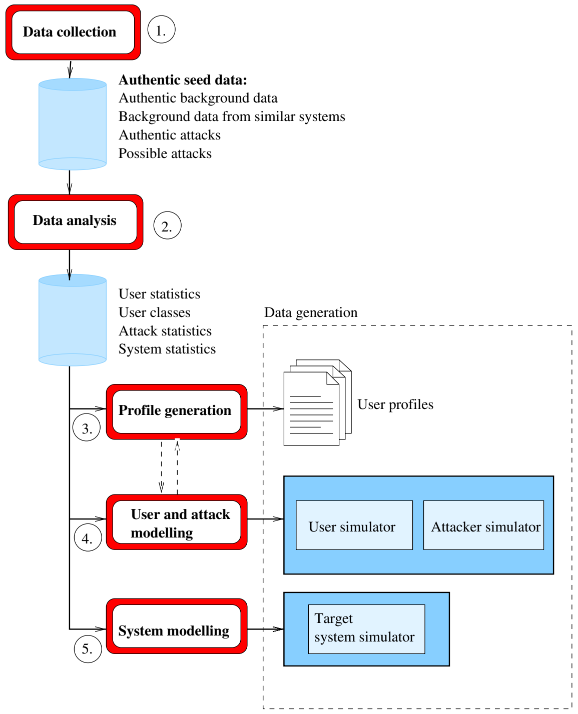

Figure 1. Synthetic data generation method

It is possible to use people instead of a user simulator to create user actions and to use the whole or parts of the real system instead of a system simulator. This may be preferable in some situations, e.g. if the system or user behavior is very complex and needs to be modeled in great detail. In our experiments, we used humans to mimic fraudulent behavior and automata to generate normal background data.

4. Authentic data

This section describes how we implemented the collection of authentic data for the VoD service. The next section (Section 5) presents the work of designing, implementing and using the data analysis, user and system model.

4.1. The target Video-on-Demand system

The VoD service was in pilot operation and had a limited number of users. It could thus provide only small amounts of log data. This system consisted of a number of components, shown in Figure 2. Each user had a set top box (stb) at home, which was connected to the Internet via a fast xDSL connection. When the set top box was turned on, it automatically contacted the service providers DHCP server (Dynamic Host Configuration Protocol server) to get a dynamic IP address. Then, the user could contact the application server, login to it, browse the video database and order a movie. The application server generated an authentication ticket for the user. The VoD server then started delivering the chosen movie after verification of the ticket.

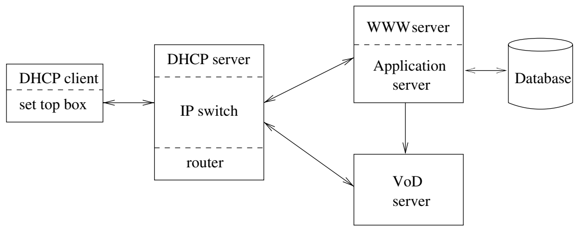

Figure 2. VoD system components

In the VoD-system, a total of 12 ”test users” were active during the service development. Thus, the amount of data generated was not sufficient for training the fraud detection modules. We used these twelve users as an approximation of ”normal” users. They had no knowledge about the implementation of the service or of the fact that their behavior would be used for synthetic data generation.

4.2. Collection of the authentic seed data

We had the opportunity to work together with the development team for the VoD service and could therefore specify the information collected in the log files. To be able to decide what information to collect, we created a database with expected frauds and the indicators or features needed to detect them. Each indicator was analyzed to find out what type of log data was needed to catch the indicator.

The test users of the VoD system were considered ”friendly users” and had been selected as test pilots based on their physical location. All users were known to the development team and thus no frauds were expected to occur. These users generated the background data we collected.

We used employees that acted according to descriptions of the expected fraud cases we had created. These fraud cases were "injected" into the system by using two dedicated set top boxes in the same way as we expect a fraudster to use it.

Data were collected over a period of about three months and the total amount of data was 65 MBytes.

4.3. Fraud data

Four fraud cases were "injected" into the authentic data:

Break-in fraud: The fraudster has "taken over" the identity of a legal user by hacking the user's set top box. The real user may use the service without knowing that he had a break-in. An indication of a break-in could be a sudden excessive usage of the service.

Billing fraud: To avoid paying for the movies ordered, the fraudster has hacked the billing server. In our experiments this was done by removing billing records in the application server log file.

Illegal redistribution fraud: This means that a customer receiving a movie in the VoD service transmits it to other people not paying for the service. Our case of illegal distribution was performed by uploading large files to a computer on the Internet some time after the completion of a movie download. Illegal redistribution of content could also be an indication of an external break-in in the set top box where the attacker in turn uses (and downloads) videos at the expense of a legitimate paying customer.

Failed logins: Several failed login attempts were added with "dummy" user ID's to imitate the behavior of people trying to guess passwords.

4.3.1. Example of a fraud case. This subsection gives an example of the process we used for each fraud case to decide what the fraud indicators were and thus what information we needed in the log files.

Fraud indicators for the fraud case *illegal (re)distribution of services* may be:

- The ratio between transmitted and received data is suspiciously high.
- A great deal of data is transmitted (some period of time) after data has been received.
- A great number of downloads are done.

For the first indicator we need user ID, IP address, bytes transmitted and bytes received in the log data. The second indicator requires IP address, bytes received, bytes transmitted and time stamp. For the third indicator we need user ID, IP address, time stamp, session ID, billing data and bytes received.

Thus, we found that the following information needed to be in the log data: (1) Data from the DHCP server containing information about when a user has his set top box switched on and off; (2) Router statistics in which the number of bytes to and from a user is registered; (3) Data from the application server containing time stamp, user ID, IP address, session ID etc. of user logins; (4) Detailed information about movie orders from the application server; (5) Billing information generated by the billing system; (6) VoD server information about what content was actually delivered to a certain user. These raw data records from the service components were converted to a common format.

5. Generation of synthetic data

The goal of the synthetic data generation was to obtain enough data of sufficient quality to be able to develop and train a fraud detection prototype for a VoD service.

5.1. Data analysis and profile generation

The seed data were analyzed and we identified a number of parameters we considered important, which we then included in the user profiles. The importance of the parameters was evaluated by studying the features needed to detect the identified template fraud examples. We also selected the parameters necessary to determine statistical values for the transitions in the state machine used to model user behavior.

The idea was to sort users into a number of classes based on their behavior. Each class of users has one set of parameter values. Because of the limited number of authentic users, it was difficult to find natural user classes without letting each user form its own user class. This, and the project time limits, resulted in the grouping of all normal users into a single class, which is a coarse simplification of normal user behavior. The frauds affecting user behavior were assigned to their own classes. Normal user behavior and fraudulent user behavior can both be scaled up during the generation process. This allows us to vary the ratio between fraudulent and normal data for our test cases.

A *perl* program parsed the log files and collected data for each transition in separate files. Small *matlab* programs plotted these data and fitted different probability distribution functions to the data. In our first data generation attempt we used the normal probability distribution function to model the time intervals for the transitions. It was obvious when the data were plotted that most of our data sets had the shape of an exponential or gamma distribution function. A chi-square test was used to check which distribution had the best fit. An example of a plot of transition time and the corresponding probability distribution function is shown in Figure 3. This histogram shows the authentic transition times from login to movie order, and the dashed line shows the corresponding probability distribution, from which we get the simulated transition times.

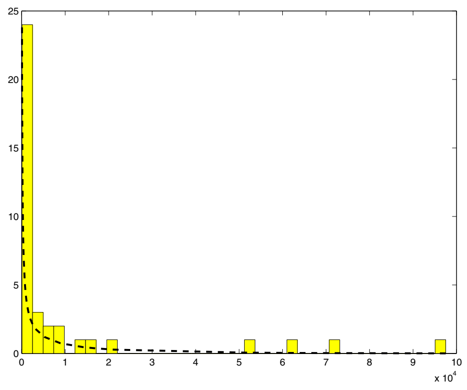

Figure 3. Histogram of time from login to order (in seconds)

The statistical profile contains the probability for each transition and a distribution function with fitted parameters for the transition time.

Statistics needed for the simulation of the system were also analyzed. For example, traffic for different events, the behavior of the DHCP server and delays were studied. These statistics were collected in a configuration file for the system simulation program.

5.2. Generation of synthetic data

The implementation of the user and system simulation is a trade-off between simplicity and realism. This process may be iterative, where a simple version is first implemented and validated to identify whether a sufficient number of the properties considered important are preserved. If this is not considered good enough, some level of detail can be added.

In our experiments, both the user and system models were implemented as an event-driven simulations in *perl*, requiring about two thousand lines of code each.

User simulation. The goal of the user simulation is to produce a chronological list of actions for each simulated user. What makes these actions realistic is the accuracy of the statistical profiles used as input and level of detail in the user model implementation. Our simulation used a finite state machine to mimic user behavior (Figure 4).

In state 0, the user's set top box (stb) is switched off. The only thing he can do in this state is to turn the set top box on and enter state 1. In state 1, he can try to log in. If it is a successful login, he is transferred to state 2, where he can start to order movies. If he fails to log in, he is transferred to state 4, where he can continue to make failed logins until

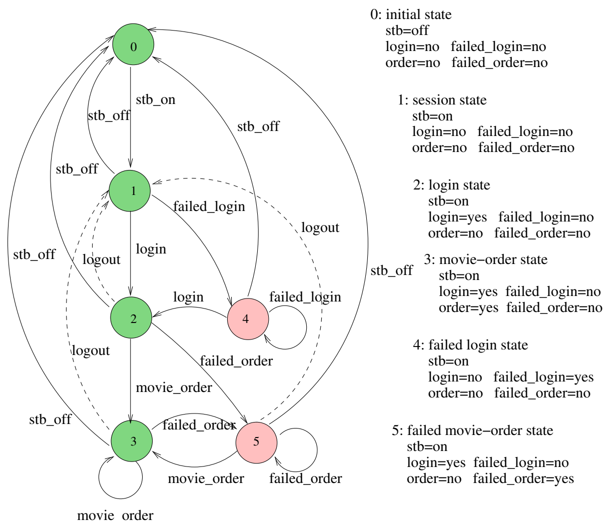

Figure 4. User state machine

he succeeds or gives up and switches the set top box off. In state 2, the user can make movie orders. If he fails he is transferred to state 5, where he can continue to try to order movies until he succeeds, logouts, or switches the set top box off. This models the actions that affect output log data in the service. The statistics derived in the analysis of authentic data are used to set the probabilities of transitions to different states and the time between the entering of one state until a certain transition is done.

Fraudulent users do not always follow this model. For example, the break-in fraudsters skipped the login procedure and were able to order movies without successful authentication. This required some extra parameters and "fraud flags" in the user simulation program. Other types of deviant behavior may require further additions to the program, even though this was the only modification we found necessary in these experiments to mimic the behavior of the injected frauds.

With this simple automata we could simulate a large number of users and attackers covering extensive periods of time. Despite its simplicity it provided sufficiently detailed user actions for the next step, system simulation.

Video-on-demand system simulation. The actions of the virtual users serve as input to the simulated target system to generate synthetic data. We used statistics from the authentic data (e.g. data volumes and delays in networks and components) to configure the simulation. A further option was added to control the generation of billing records in order to simulate fraudsters that circumvent the billing subsystem.

We used an event-driven simulation. Each user action in the input data was first fed to the set top box module. From there it was added to the simulation list to be passed on to the right component module after a configurable delay. The possible user events were *turn on set top box*, *turn off set top box*, *login*, and *movie order*. Figure 5 illustrates the resulting event flow in the simulated system when it is fed with a login action. The login event is passed from the set top box to the access node (IP switch) via the simulation list ("sim" in the figure). From the access node, the event is passed to the router module, which calculates router statistics for the event, and also to the application server, where the login is done. The success of the login is determined during the user simulation and thus included in the input user action. The application server generates an authentication log entry (AuthenticationNotification) of the same format as that of the real system. In this way all events are passed around in the system, and the proper components generate log messages when events arrive.

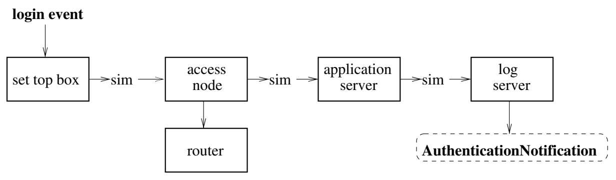

Figure 5. Flow in simulated system at a login action

Our simulation model is a coarse approximation of the real system environment. We do not model network behavior at packet level, for example, as the router statistics are only measured over a time interval. Other shortcomings are described in Section 8.

6. Input data for the detection experiment

We verified the generated data by using them to develop and train a fraud detection prototype, i.e. according to the first application area mentioned in Section 1. This section describes the data used in the experiment.

6.1. Authentic data

The authentic data needed for the detection experiments are the data created by authentic fraud users. These users have long periods of rather normal behavior and occasional fraud sessions. The frauds are described in Section 4.3. Since we had very few authentic fraud sessions, we had to use the same data set here as we used as seed data for the synthetic data generation process.

6.2. Generated synthetic data

The simulation described in Section 5, resulted in synthetic data for seven months containing six hundred "normal" users. Three fraud types, out of the four described in Section 4.3, were simulated. We used 100 break-in fraudsters, 100 fraudsters doing illegal redistribution, and finally 100 cases of fraudsters that hacked the billing server.

This resulted in several GBytes of synthetic raw log data in unpacked text format. The raw log data were converted to a common format where data were grouped into sessions in the same way as the authentic log data were converted. This resulted in about 80 MB of log data for each group of 100 users. It was possible to generate log data in less than a day on a normal PC for 100 users acting under a period of seven months.

6.3. Subjective assessment of data

To form a subjective concept of the simulated data and try to estimate its validity, we did a great deal of visualizations of the authentic and synthetic users. Below we show some of the plots produced in this process.

Normal users. The plots show the events for a user for a period of 15000 minutes (about 10 days). The first two plots (Figure 6 and 7) show the behavior of two different authentic users. The next plot (Figure 8) shows the behavior of one of the synthetic users.

The scale of the x-axis is in minutes (from the start date of the log files). The y-axis shows which state the user currently is in. These numbers correspond to the states in the user model (see Figure 4). The dotted red line (or grey if viewed in black-and-white printing) follows the user state.

level 1.0: User has switched on the set top box
level 2.0: User is logged into the application server
level 3.0: User has ordered a movie
level 4.0: User has made a failed login
level 5.0: User has made a failed movie order
Finally we use "level 0.5" to show downloading or uploading sessions (does not correspond to any user state).

The fourth and fifth levels are used less frequently since these correspond to failed logins and failed movie orders, which are not common. The events that cause the user to switch states are plotted as spikes in the graph, with the height of the state the user is entering, i.e. logins with a height of 2.0 and orders a height of 3.0.

The user in Figure 6 is very active. His set top box is switched on most of the time, several days in a row. The plot shows two sessions. In the first session he waits a short time before logging in. After some hours he orders a movie. Shortly after the movie order, download starts and he continues to download data for several hours. Then the session continues with several more logins and downloads. It seems that the user is logged out automatically after some time, but this is not possible to see in the log files. Downloads often start at login, and downloads are done even when there are no movie orders. The reason for this is that users can also watch free TV channels, and that they probably watch movie trailers before ordering a movie.

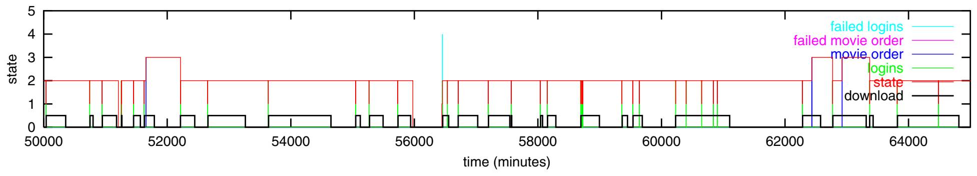

Figure 6. Plot of authentic user (user ID 0.123)

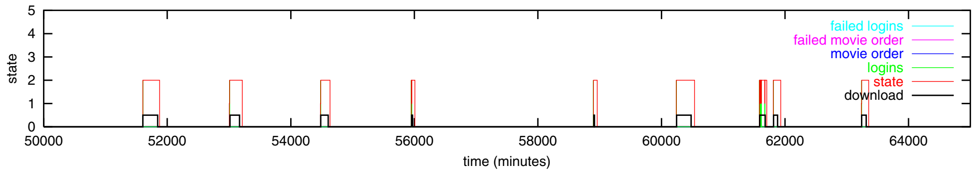

Figure 7. Plot of authentic user (user ID 0.128)

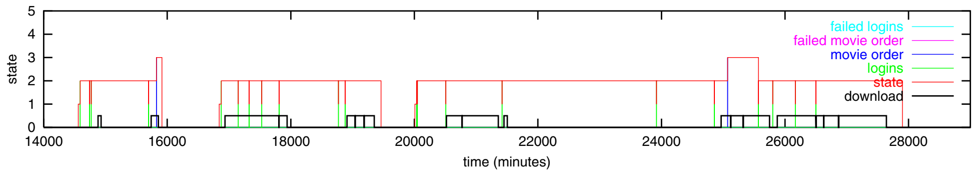

Figure 8. Plot of synthetic user (user ID 2.137)

On the opposite end, the authentic user in Figure 7 is not an avid user of the service. He has several shorter sessions but no movie order. Still, downloading is going on during the sessions. The short sessions indicate that he always switches the set top box off when not watching TV. Here the downloads starts regularly, and a detailed examination revealed that all these sessions occur in the evenings, lasting less than a couple of hours. The conclusions that can be drawn from these plots are that the first user seems to be home during daytime, he has irregular TV habits, and leaves the set top box switched on even when it is not used. The second user probably has a more ordered life, works during daytime, watch TV in the evenings, and sleep at night.

Since we use only one normal user profile in the simulation, the synthetic user in Figure 8 should behave like a medium active user , which seems to be the case. Hence, this synthetic user also has very similar behavior to the other synthetic users. He has rather long sessions, but not as long as the first authentic user. He also orders fewer movies and downloads data less often. However, it seems that many of the download sessions are comparatively long, which may be an implementation mistake in the simulation.

Fraud users. An example of a break-in fraudster is shown in Figure 9. In this figure, the fraud behavior is not obvious. However, in the next Figure (10), only the logins and orders are shown for the break-in fraud user. Here we can see that there are orders in the sessions before the user has logged in. Comparing this behavior to that of a normal user, we see that there is always a login before an order in a normal session.

The billing fraud plots looks very much like those of normal authentic and synthetic users. The absence of billing records cannot be seen in these plots, and therefore we do not show the plots here.

A mistake was made in the realization of the illegal redistribution fraud in the authentic data which caused the router log file to be empty during this period. This meant that we did not have complete authentic seed data for this type of fraud. Since we wanted our detection system to be able to detect also this type of fraud, which was considered an important type, we decided to “manufacture” an illegal redistribution profile manually. The profile reflected the planned behavior, i.e. it contained traffic download after orders and a great deal of upload traffic some hours after the download. We used this to perform detection tests using a second set of data. However, since we could not test this case against authentic data our results are of limited value and we will address this fraud case further in this report.

It should be noted that all types of frauds that we wish to trigger our detection mechanism can be created “manually” in this way, by editing the statistical profile. This can also be used for testing the detection capability of a fraud detection system when frauds or background traffic are varied.

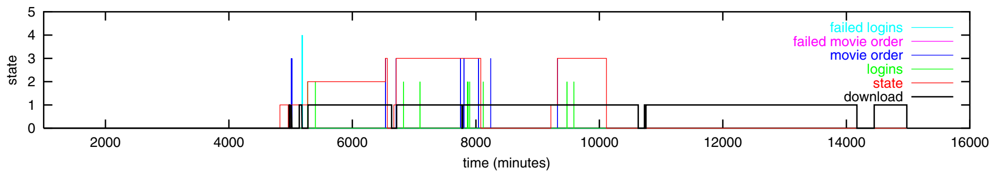

Figure 9. Plot of break-in fraudster in authentic data (user ID 0.131)

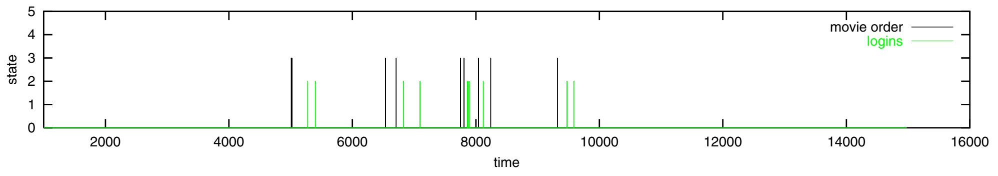

Figure 10. Plot of break-in fraudster showing logins and orders (user ID 0.131)

7. Detection experiments

The fraud detection system used in the experiments was a neural network with an added ability to handle temporal dependencies. The neural network was trained with synthetic data containing about 25% fraudulent and 75% normal behavior. The detection capability of the neural network was then tested with authentic data containing frauds. The detection results with authentic data were compared to detection results with test sets created from the synthetic fraud data to find out how much they differ.

7.1. The neural network

The neural network was trained using synthetic data and was then used to detect the attacks existing in the authentic data. A feed-forward neural network model was used, consisting of seven inputs, a single hidden layer with seven nodes, and a single output giving indications of fraud. The net was trained independently for each fraud type, thus requiring only a single output bit. The initial input weights were randomized before training to prevent the network from finding local minima. The neural network model was implemented in C with approximately 900 lines of code.

As conventional neural network architectures and models are not well suited for patterns that vary over time, an exponential trace memory was used [9]. The memory can be viewed as a buffer maintaining a moving average (exponentially weighted) of past inputs: $\bar{x}_i(t) = (1 - \mu_i)x_i(t) + \mu_i\bar{x}_i(t-1)$. The moving average is calculated for all input parameters over a configurable time interval grouping events together. The configurable μ_i allows for the representation of averages spanning various intervals of time. The higher the value of μ_i, the fewer the current input patterns influenced by input from previous intervals. We used $\mu_i = 0.7$ in our detection tests, which proved to provide a sufficient decay rate for our purposes.

7.2. Neural network input

The sum of all input events (over an interval) was fed to the inputs of the neural networks. For simplicity, we used an interval of 1440 minutes (24 hours). This allowed our detection scheme to detect fraud with a granularity of 24 hours, which should be sufficient for most service environments using fraud detection. A finer granularity could have been chosen but would not have been useful, as most users only order a few movies over a period of this length. The input events were assigned to the neural network's inputs as follows:

1. Sum of successful login attempts
2. Sum of failed login attempts
3. Sum of successful movie orders
4. Sum of failed movie orders
5. Sum of movie delivery notifications
6. Sum of billing notifications
7. Ratio between uploaded and downloaded number of bytes [(dl/(1+ul))/1000]

The values were not normalized as most input values were roughly within the same range. An exception was input 7, which was divided by 1000 to fall within a range of magnitude similar to that of the other input values.

Parsing audit records, handling intervals and preparing audit data according to the network inputs were done using a *perl* script (approximately 500 lines of code).

7.3. Detection of billing fraud

Synthetic data containing occurrences of billing fraud (see Section 4.3) were used to train the neural network. Thirty-five synthetic users active in a total of 3,000 (1,440 minutes) intervals were randomly selected. Of these intervals, 2,023 were non-fraudulent and 976 contained fraudulent activities. In the plots below, we define *the epoch* as the time interval containing the first available log entry. The network was trained for 1,000 rounds, selecting input values from random intervals. One thousand rounds of training took approximately 15 minutes on 350 MHz Silicon Graphics O2+ having 650 MB of memory.

Figure 11 shows detection tests using a second set of synthetic data. Intervals containing fraudulent activities were successfully detected with few errors. A few occurrences of *false negatives* were found, which were caused by periods of user inactivity, i.e. the user did not order any movies during the interval, which could be perfectly normal even for a fraudulent user. In addition, at the end of the fraud periods, *false positives* were observed for one to two intervals. This was caused by the memory that handles temporal correlation of data.

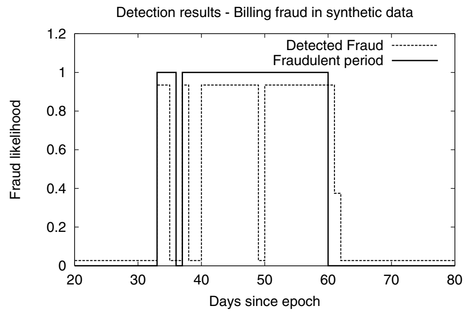

Figure 11. Detection of Billing fraud in synthetic data

If a significant number of events are processed during a few intervals, the trace memory could contain enough history data to trigger an alarm. This should not pose a problem in most cases, as the error follows a period of actual fraud. The value of μ_i, controlling the decay rate of the trace memory, plays a role in controlling false positives and negatives. A fast decay rate would lead to more false negatives while a slower decay rate would lead to a higher number of false positives at the end of fraudulent periods.

Next, the events contained in the authentic data (with labeled occurrences of fraud) were fed to the network. The data were known to contain billing frauds for a single user between May 18 and June 14 (day 10 - day 30 in Figure 12). The results are shown in Figure 12, which clearly shows that our neural network detection model works as trained.

The problem of false positives at the end of a fraudulent period remains, but does not prevent a security officer from successfully identifying the occurrence of those fraudulent activities, which can then be further investigated.

7.4. Detection of break-in fraud

The break-in frauds mimic a user who breaks into a customer's set top box and seriously alters the user's usage

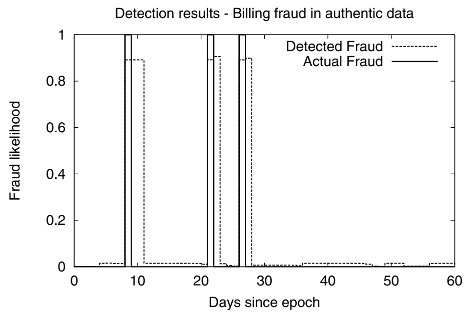

Figure 12. Detection of Billing fraud in authentic data

behavior (e.g. by ordering a substantially higher number of movies over some periods). As previously mentioned, break-in frauds were simulated by a team of ”fraudsters”, and from their actions synthetic data containing fraudulent users were generated and used for training and testing. Again, 35 synthetic users were chosen that were active in a total of 3,000 intervals. Of these, 2,016 were nonfraudulent and 984 contained fraudulent activities. Similar to the training in billing fraud, the network was trained for 1,000 rounds with random input values.

Figure 13 shows the detection results of a second set of synthetic data. A nearly perfect match is achieved in the interval shown. However, two intervals show false negative behavior, one in the beginning of the fraudulent period and one at the end. Again, this does not seriously affect the usefulness of the detection system.

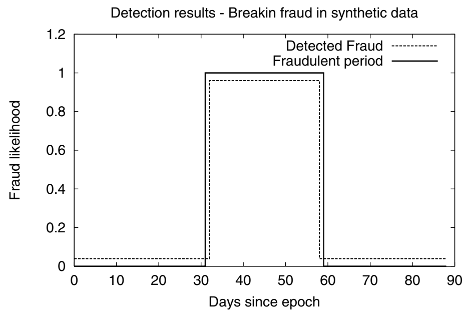

Figure 13. Detection of break-in fraud in synthetic data

The detection of break-in frauds in authentic data is shown in Figure 14. The result is not as promising as in the synthetic data. In addition to the period of actual fraud (days three to six), several false positives are shown at various intervals (approximately at days 33, 62, 68, 71, 75, 81, 85). An investigation of the reason for this showed that the fraudulent user did not deviate a great deal in consumption compared to the "normal" user. This made it difficult for the neural network to distinguish between normal and fraudulent behavior.

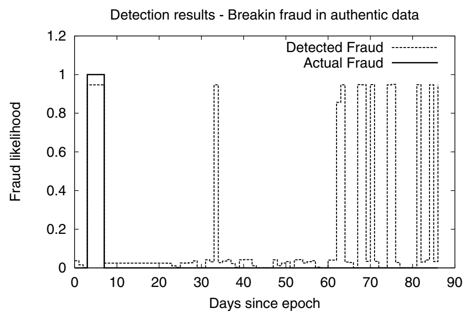

Figure 14. Detection of break-in fraud in authentic data

The average numbers of movie orders per day are plotted in Figure 15. The fraudulent period (day 4-7) did not exceed forthcoming consumption enough to allow the network to successfully differentiate fraudsters from normal users. Clearly, our team of "fraudsters" did not succeed in their job of ordering an excessive amount of movies during that time period. The false positives were also affected by the temporal memory, as the false alarms were preceded by quite a few days of moderate usage which together, over time, built up sufficiently large input values for the neural networks to trigger an alarm. Once again, this illustrates the value of carefully balancing the decay rate of the memory function for each type of fraud.

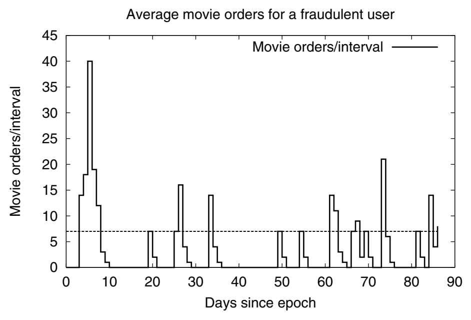

Figure 15. Movie order averages for a fraudulent user (authentic data)

7.5. Quantitative results

Table 1 shows the detection results. The *Sensitivity* and *Specificity* are shown in the first two columns of the table. *Sensitivity* [true pos / (true pos + false neg)] shows to which extent fraudulent activity is classified correctly. If detection is high in false negatives, the sensitivity becomes poor. The *Specificity* [true pos / (true pos + false pos)] indicates the degree of misclassification of non-fraudulent events. If a test shows high a false positive value, the specificity becomes poor. A goal of classification systems is to be high in both specificity and sensitivity. The *Number (#) of periods* defines the total time frame during which detection was performed. In our tests, a single time quanta was 1440 minutes. The periods specified for each test are harmonized with the graphs illustrated in Figures 13, 14, and 15. For each fraud type, *True positives, False Positives, False negatives* and *True negatives* are shown.

As can be seen in the table, *specificity* is somewhat better for the synthetic test data than for the authentic data. This is expected as the neural network detector was trained using synthetic data and also these data are more regular. However, the *sensitivity* is better for the authentic data, which is more unexpected. This is likely a result from the fact that the synthetic data had a higher fraud rate. In short, the authentic data contained mostly of normal data with only short periods of fraud, while the synthetic data had a significantly higher percentage of fraud, which provided more opportunities for misclassification. Overall, we believe that the differences between synthetic and authentic data are reasonably small and indicate that the training using our synthetic data was successful.

8. Discussion of results and future work

Scalability versus complexity tests. We created hundreds of simulated users acting for a period of seven months. This was sufficient to train and test the fraud detection prototype. We are thus satisfied with the ability of the method and the implementation to scale the number of users and the time period of the synthetic logs.

An interesting scalability issue that should be studied further is the effects of a more complex modeling of the users and the system. Implementing outliers and several user classes should not affect the scalability very much, but the use of a more complex state machine for user behavior would probably affect the performance of the simulation tool.

Diversity of background data. The background data were rather homogeneous. There were, for example, no "outliers" among the users. One reason for this was that only one user profile was used, which meant that all the simulated users behaved according to the same statistical distributions. The obvious solution to this problem is to use several user classes with different profiles, which was our initial intention, but time limits in the project prevented this. For our detection experiments, it seems that the diversity of the background data was good enough for the billing fraud, since the normal behavior in the authentic data did not trigger very many false alarms. However, it posed a

Table 1. Detection results

Fraud	*Sensitivity*	*Specificity*	*# of periods*	*True pos.*	*False pos.*	*False neg.*	*True neg.*
Billing fraud (synthetic data)	0.89	0.98	365	97	3	11	254
Billing fraud (authentic data)	1.0	0.93	61	3	4	0	54
Break-in fraud (synthetic data)	0.92	1.0	89	26	0	2	61
Break-in fraud (authentic data)	1.0	0.86	87	4	11	0	72

bigger problem for the break-in fraud, where normal users and fraudsters showed very similar behavior. It would be interesting to make tests using background data with different degrees of homogeneity.

The user model. The user model may be too simplistic for certain applications. Some behavior was deliberately left out of the model to keep it simple, such as use of the network for other things than downloading movies. Neither were long-term variations in users' activity modeled. It would be interesting to develop a more advanced user model. However, we believe that the user model is of sufficient detail in the VoD application.

The system model. We used static parameters for controlling delays and some of the router statistics in the system model. In a real system, these would vary depending on load etc. Future versions of our simulator will be more dynamic in simulating such dependencies. Neither did our simulator model distinguish "strange" user behaviors, such as malformed network traffic, spoofing etc, which limits the ability to simulate attacks based on bugs in the software and the hardware. This could also be improved in future versions of the simulator.

Fraud cases. Our process of injecting frauds in authentic data can be further improved. We injected frauds in only a few users' behavior and for only short periods of time. This was acceptable for our application but makes it more difficult to verify that the modeling of the users and the system is sufficient for more general use and other target applications. We plan to establish a more lengthy list of fraud scenarios for future experiments.

Suitability for other types of services. Future work will show whether the data generation process works equally well for other types of services and for intrusion detection.

9. Conclusions

We have developed a method for generating large amounts of synthetic log data that preserve statistical properties of a selected set of authentic data used as a seed. We have experimentally shown that the synthetic data generated can be successfully used for training and testing a fraud detection system. Future experiments will verify whether this also holds for more general classes of seed data and for other types of fraud detection systems. We learned several lessons in the process of generating and testing data which will help us to further improve our methodology.

10. Acknowledgments

We would like to thank Telia Research AB (nowadays TeliaSonera) and the people working on the VoD service for their cooperation and help. We would also like to thank the participants of the EURESCOM project ”P1007”.

References

[1] P. Burge, J. Shawe-Taylor, Y. Moreau, B. Preneel, C. Stoermann, and C. Cooke. Fraud detection and management in mobile telecommunications networks. In *Proceedings of the European Conference on Security and Detection ECOS 97*, London, April 1997. ESAT-SISTA TR97-41.

[2] P. K. Chan, W. Fan, A. L. Prodromidis, and S. J. Stolfo. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems*, 14(6), Nov/Dec 1999.

[3] 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, Mar. 1998.

[4] J. W. Haines, R. P. Lippmann, D. J. Fried, E. Tran, S. Boswell, and M. A. Zissman. 1999 darpa intrusion detection system evaluation: Design and procedures. Technical Report Technical Report 1062, MIT Lincoln Laboratory, Feb. 2001.

[5] H. Kvarnström, E. Lundin, and E. Jonsson. Combining fraud and intrusion detection - meeting new requirements. In *Proceedings of the fifth Nordic Workshop on Secure IT systems (NordSec2000)*, Reykjavik, Iceland, Oct. 2000.

[6] W. Lee and D. Xiang. Information-theoretic measures for anomaly detection. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, May 2001.

[7] E. Lundin, H. Kvarnström, and E. Jonsson. A synthetic fraud data generation methodology. In *Lecture Notes in Computer Science, ICICS 2002*, Laboratories for Information Technology, Singapore, Dec. 2002. Springer Verlag.

[8] R. A. Maxion and K. M. Tan. Benchmarking anomaly-based detection systems. In *International Conference on Dependable Systems and Networks*, New York, New York, June 2000. IEEE Computer Society Press.

[9] M. C. Moser. *Neural net architectures for temporal sequence processing.* Addison-Wesley Publishing, Redwood City, CA, 2001.

[10] N. J. Puketza, K. Zhang, M. Chung, B. Mukherjee, and R. A. Olsson. A methodology for testing intrusion detection systems. *Software Engineering*, 22(10), 1996.

[11] K. M. C. Tan and R. A. Maxion. Determining the operational limits of an anomaly-based intrusion detector. *IEEE Journal on Selected Areas in Communication*, 21(1), Jan. 2003.