Two types of Intrusion detection schemes –

1. Misuse – exploit the system vulnerabilities
2. Anomalous behavior of use – unusual behaviour of the user of a system

Difficult to build a model that will accept legitimate behaviour as novel legitimate use of the of the system.

It is difficult to anticipate all possible variations in behaviour, but can be tracked in 3 ways –

1. Instead of general legitimate use, the behaviour of individual user or interaction to the system can be modeled. Chracterising behaviour for individual is easier than to do it for all other users simultaneously
2. Patterns of behaviour can be learned instead of having them to describe by handcoding
3. Detecting intrusion realtime, as user is typing commands order of commands vary a lot. It is enough to recognize distribution of command over entire login session or for entire day, differs form usual.

NNID based on above three ideas

* Backpropagation NN, to identify users based on what commands they use during the day.
* SYS Admin runs NNID at EOD to see session matches with normal pattern
  + Or Investigate the matter for unsual behavior
* NNID implemented in UNIX env and consists of keeping logs of commands executed
* Histogram for each user, learning user profiles, matches an audit record with the appropriate profile, pdates the profile whenever necessary, and re- ports any anomalies detected.
* 96% accurate in detecting anomalous behaviour, false alarm – 7%.
* Learning offline Monitoring system can active better performance
* Than to detect anomalies online in command sequence with less effort

Many misuse and anomaly intrusion detection systems (lDSs) are based on the general model proposed by Denning (1987).

General models are based on –

* independent of the platform,
* system vulnerability
* type of intrusion
* a rule set, is used for detecting misuse

Rule based statistical IDS models

* Often Statistical methods are used based on distributed subject behavior capture, such approaches require that the distribution of subjects' behavior is known.
* The behaviour can be represented as Rule based model
* behavior can be represented as a rule-based model (Garvey and Lunt 1991),
  + in terms of predictive pattern generation (Teng et al. 1990), or
  + using state transition analysis (Porras et al. 1995)
* Pattern matching techniques are then used to determine whether
  + the sequence of events is part of normal behavior,
  + constitutes an anomaly,
  + fits the description of a known attack.

IDS can be Online or Offline

* Off-line IDSs are run periodi- cally and they detect intrusions after-the-fact based on system logs.
* On-line systems are designed to detect intrusions while they are happening, thereby allowing for quicker intervention.
* Online requires high computation power since they are continuously monitoring
* Decisions are made quickly hence cannot be reliable

Drawback of Online - most of the effort goes into predicting the order of commands

NNID System

The NNID anomaly intrusion detection system is based on identifying a legitimate user based on the distribution of commands she or he executes.

Explanation - his is justifiable because different users tend to exhibit different behavior, depending on their needs of the system. Some use the system to send and receive e-mail only, and do not require services such as programming and compilation. Some engage in all kinds of activities including editing, programming, e-mail, Web browsing, and so on.

* Use of Program - even two users that do the same thing may not use the same application program. some may prefer the "vi" editor to "emacs", favor "pine" over "elm" as their mail utility program, or use "gcc" more often than "cc" to compile C programs.
* Frequency of command use 0- he frequency with which a command is used varies from user to user.
* The set of commands used and their frequency, therefore, constitutes a 'print' of the user, reflecting the task performed and the choice of application programs, and it should be possible to identify
* this approach works even if some users have aliases set up as short- hands for long commands they use frequently, because the audit log records the actual commands executed by the system.
* Users' privacy is not violated, since the arguments to a command do not need to be recorded. That is, we may know that a user sends e-mail five times a day, but we do not need to know to whom the mail is addressed.

Phases to build NNID –

1. Collecting training data: Obtain the audit logs for each user for a period of several days. For each day and user, form a vector that represents how often the user executed each command.
2. Training: Train the neural network to identify the user based on these command distribution vectors.
3. Performance: Let the network identify the user for each new command distribution vector. If the network's suggestion is different from the actual user, or if the network does not have a clear suggestion, signal an anomaly.

Experiments –

NNID system was built and Tested on machines assigned to a research group at Department of Electical and computer Engg at univ. of Texas at Austin.,

1. The operating system (NetBSD) provides audit trail logging for accounting pur- poses and this option had been enabled on this system.

2. The number of users and the total number of commands executed per day are on an order of magnitude that is manageable. Thus, the feasibility of the approach could be tested with real-world data without getting into scalability issues.

3. The system is relatively unknown to outsiders and the users are all known to us, so that it is likely that the data collected on it consists of nonnal user behavior (free of intrusions).

12 days, resulting in 89 user-days

decided to simply use a set of 100 most common commands in the logs (listed in Table 1), and let the network figure out what infonnation was important and what superfluous.

Intelligent selection of features might improve the results some but the current approach is easy to implement and proves the point.

INTERVALS -- introduce more overlap between input vectors, and therefore better generaliza- tion, the number of times a command was used was divided into intervals. There were 11 intervals, non-linearly spaced, so that the representation is more accurate at lower frequen- cies where it is most important. The first interval meant the command was never used; the second that it was used once or twice, and so on until the last interval where the command was used more than 500 times. The intervals were represented by values from 0.0 to 1.0 in 0.1 increments. These values, one for each command, were then concatenated into a 100-dimensional command distribution vector (also called user vector below) to be used as input to the neural network.

The standard three-layer backpropagation architecture was chosen for the neural network. The idea was to get results on the most standard and general architecture so that the feasibility of the approach could be demonstrated and the results would be easily replicable.

The input layer consisted of 100 units, representing the user vector; the hid- den layer had 30 units and the output layer 10 units, one for each user. The network was implemented in the PlaNet Neural Network simulator (Miyata 1991)

Results –

To avoid overtraining, several training sessions were run prior to the actual experiments to see how many training cycles would give the highest performance.

The resulting four networks were tested in two tasks:

1. Identifyingtheuservectorsoftheremaining4days.Iftheactivationoftheoutput unit representing the correct user was higher than those of all other units, and also higher than 0.5, the identification was counted as correct. Otherwise, a false positive was counted.

2. Identifying 100 randomly-generated user vectors. If all output units had an acti- vation less than 0.5, the network was taken to correctly identify the vector as an anomaly (i.e. not any of the known users in the system). Otherwise, the most highly active output unit identifies the network's suggestion.

Since all intrusions occur under one of the 10 user accounts, there is a 1110 chance that the suggestion would accidentally match the compromised user account and the intrusion would not be detected. Therefore, 1/10 of all such cases were counted as false negatives.

The second test is a suggestive measure of the accuracy of the system

* not possible to come up with vectors that would represent a good sampling of actual intrusions;
* here was to generate vectors where the values for each command were randomly drawn from the distribution of values for that command in the entire data set
* random test vectors had the same first-order statistics as the legitimate user vectors, but had no higher-order correlations.
* neutral but realistic sample of unusual behavior.
* All four splits led to similar results.
* e networks rejected 63% of the random user vectors, leading to an anomaly detection rate of 96%.
* legitimate user vectors 93% of the time, giving a false alarm rate of 7%.
* 24 legitimate user vectors, the network identified 22.
* he correct output unit is very highly activated, indicating high certainty of identification

each of the 24 test vectors in one of the 4 splits tested. The output units are lined up from left to right, and their activations are represented by the size of the squares. In this split there were two false alarms: one is displayed in the top right with activation 0.01, and one in the second row from the bottom, second column from the left with 0.35. All the other test vectors are identified correctly with activation higher than 0.5.

Future Work - Although there are many computer systems that have no more than a dozen users, most intrusions occur in larger systems with hundreds of users.

With more users, the network would have to make finer distinctions, and it would be difficult to maintain the same low level of false alarms. But the rate of detecting anomalies may not change much, as long as the network can learn the user patterns well. Any activity that differs from the user's normal behavior would still be detected as an anomaly.

Training the network to represent many more users may take longer and require a larger network, but it should be possible because the user profiles share a lot of common struc- ture, and neural networks in general are good at learning such data. Optimizing the set of commands included in the user vector, and the size of the value intervals, might also have a large impact on performance

Since NNID parses daily activity of each user into a user-vector, the user profile can be updated daily. NNID could then be retrained periodically. In the current system it takes only about 90 seconds and would not be a great burden on the system.

CONCLUSION

Experimental evaluation on real-world data shows that NNID can learn to identify users simply by what commands they use and how often, and such an identification can be used to detect intrusions in a network computer system. The order of commands does not need to be taken into account. NNID is easy to train and inexpensive to run because it operates off-line on daily logs. As long as real-time detection is not required, NNID constitutes a promising, practical approach to anomaly intrusion detection.

Drawback—

* Only on UNIX system
* This is offline monitoring system, but with better performance

Approaches to Intrusion Detection

Approch using NNID