GUI-Based User Profiling for Masquerade Detection in Linux Systems

Wilson Naik Bhukya, Suneel Kumar & Atul Negi

University of Hyderabad, India

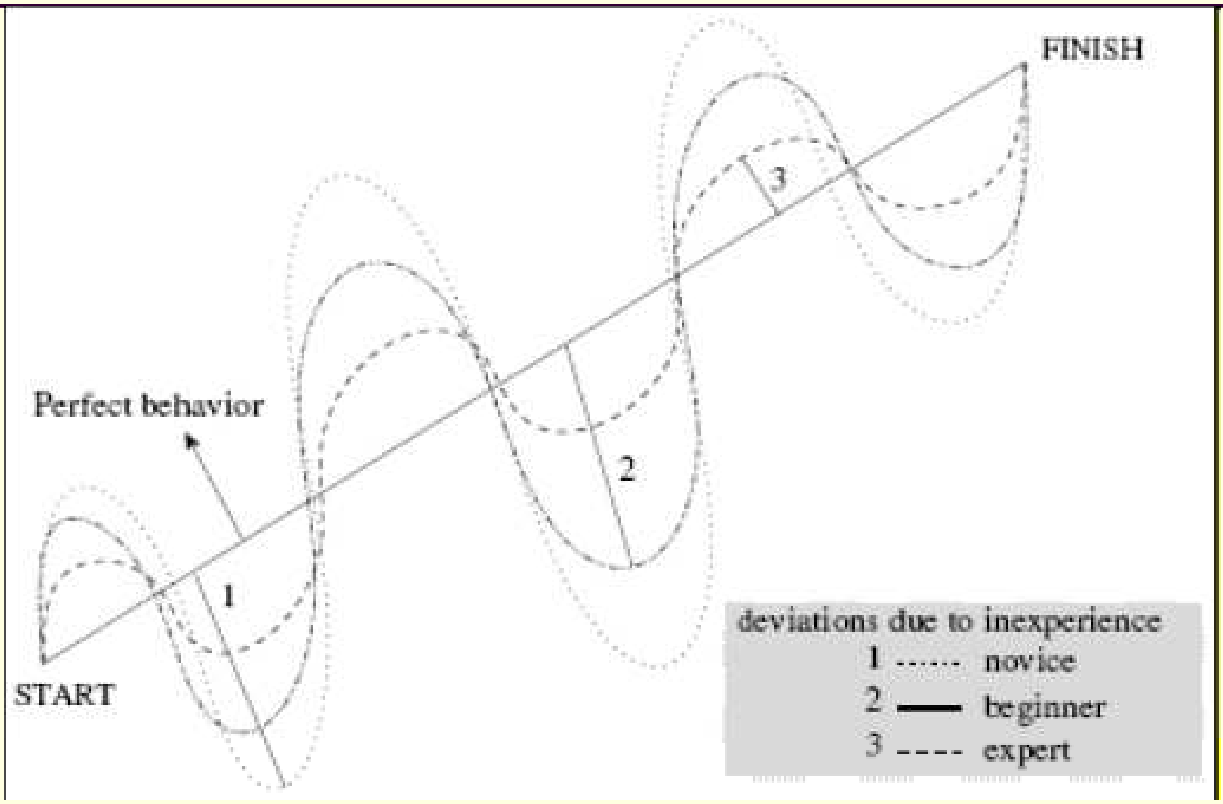
Outline

- Motivation
- Related Work
- Proposed Method
- Results
- Conclusion & Future Work

Motivation

- ✓ GUI usage Profiling which is different from command line user profiling.
- ✓ Since command line data can not capture entire user activities performed using GUI.
- ✓ A profiling technique is required to log the GUI user activities
- ✓ To my knowledge no profiling technique available in literature for Linux systems for capturing GUI events for Masquerade detection.

User Behaviors



[Garg and others 2006]

Introduction

Masquerader

- ✓ Spoofing: Impersonating other users, e.g. by forging the originating e-mail address, or by gaining password access [Denning 1997]
- ✓ Masquerader is someone who pretends to be another user while invading Target user's accounts, directories, or files (Anderson,1980)
- ✓ Five phases of general hacking [Counter hack, 2001 Prentice Hall]
  - scanning system vulnerabilities.
  - gaining access using application or OS vulnerabilities
  - gaining system administrator privilege
  - preventing detection
  - maintaining access using Trojans, backdoors or Root kit

Related work

- Attempts has been made by Schonalu and other researchers to tackle the problem of Masquerade detection [Schonalu & others]
- SEA data set which was created using *acct* utility on Unix system to capture the command line data.
- Many research papers were published using this data set .

Related Work (contd.)

- Masquerade detection using mouse movements in web pages of a browser [Pausra & Bradley, 2004]
- Masquerade Detection using process details of Windows NT [Li, Manik 2004]
- Masquerade Detection using GUI user profiling & user usage [Garg 2006] [Imsand 2007]

Related Work

- Limitations of the previous approaches
  - Only Command line data
  - Training and testing time is very high
  - Only few GUI events were captured
  - Low detection rate with high false positives

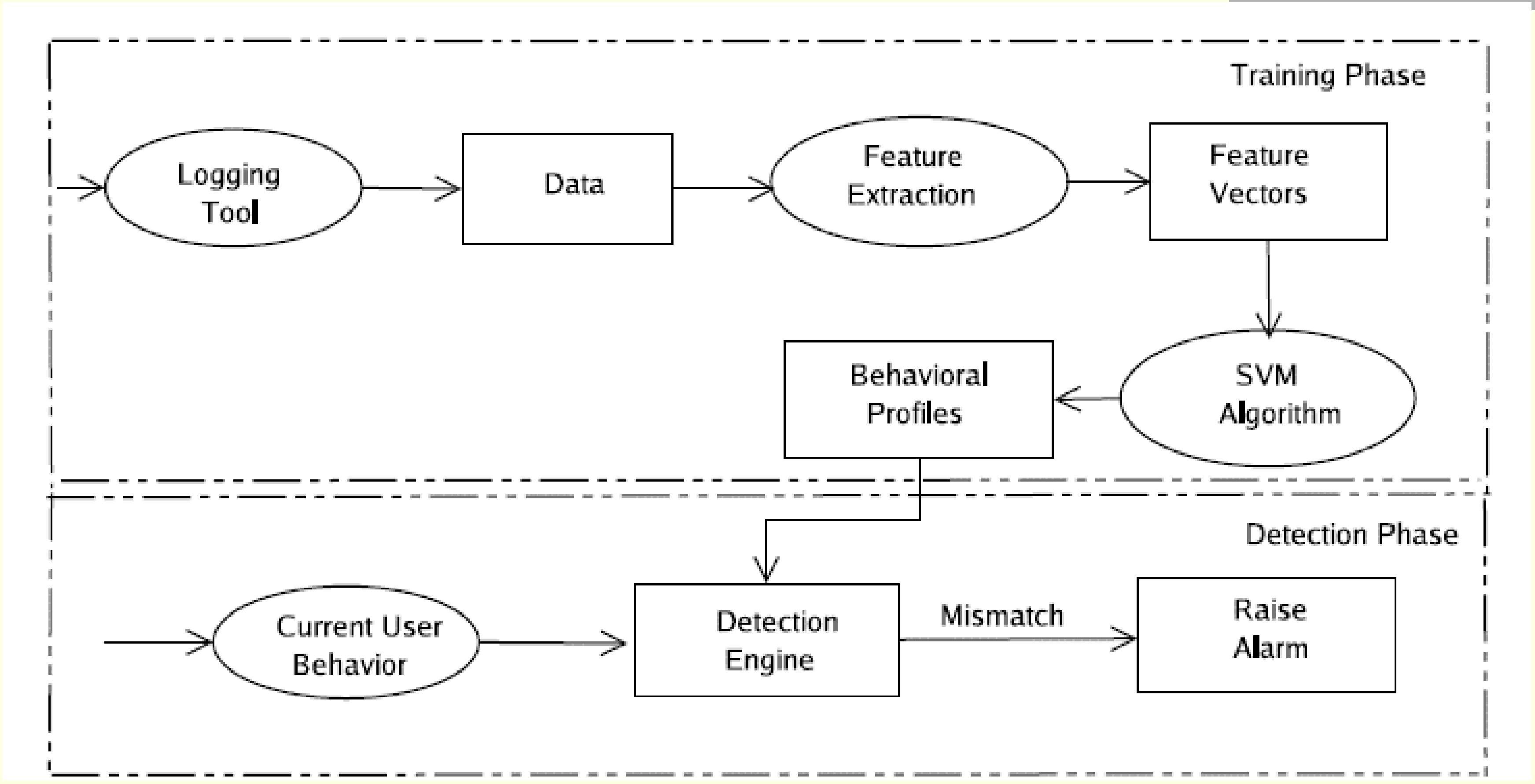
Proposed Method

- Developed a event logging tool to capture the GUI user events like mouse events and key board events.

- All these GUI events are captured for different users, for different random sessions

- Captured events are sanitized and features are extracted to train and test the SVM .

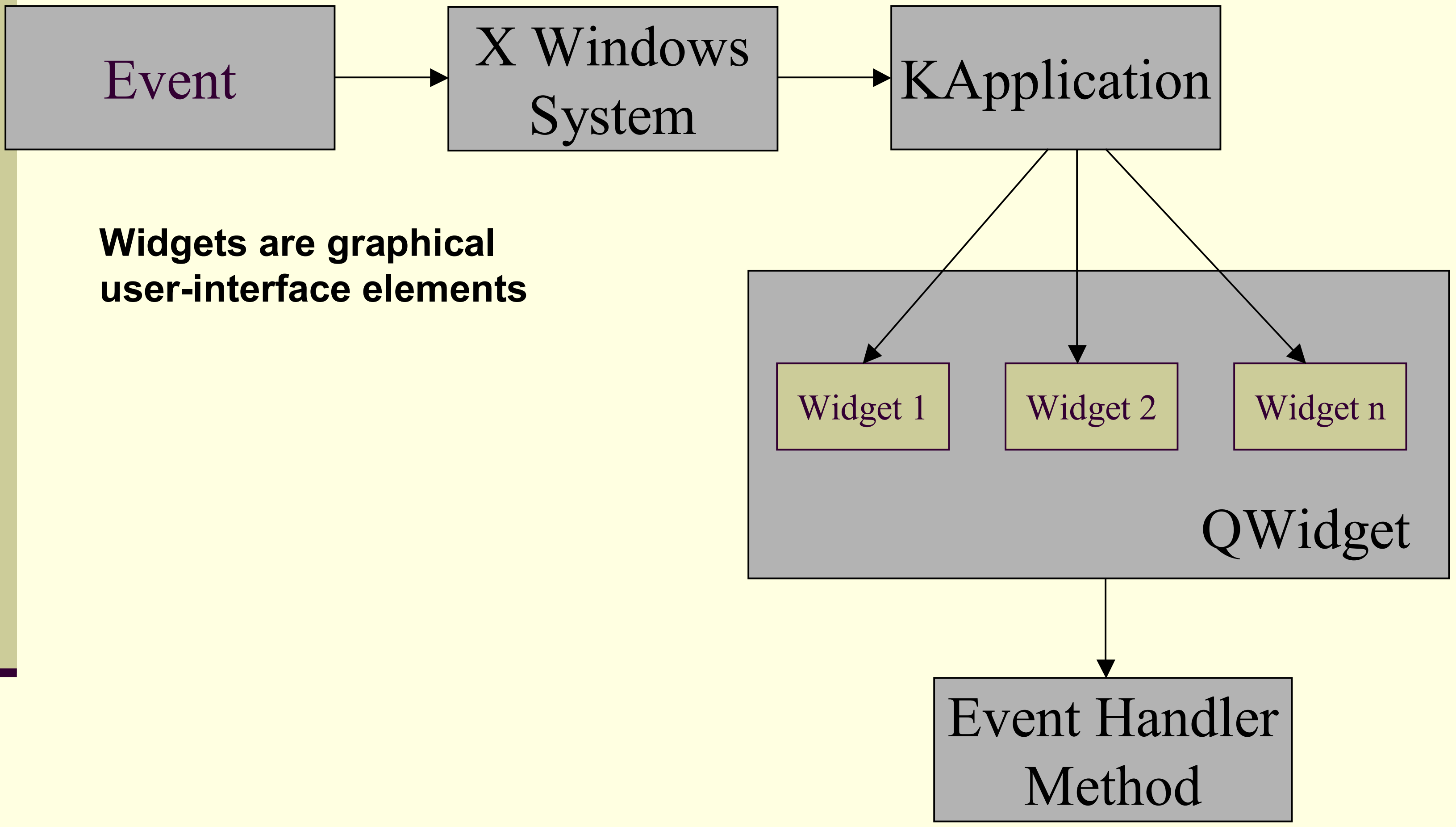
Architecture of the System



Proposed Method (contd..)

- Steps Involved
  - Event logging
  - Sanitization
  - Feature extraction
  - Training and testing on SVM classifier

Event capturing



Widgets are graphical user-interface elements

Event Logging

- QWidget handles window system events, manages generic widget attributes, knows about its parent and children.
- Widgets use signals to communicate user interaction and/or changes in their state.
- A pushbutton, for example, might emit a clicked() signal to indicate that the user has clicked the button.
- Imagine an FM radio application: You connect the clicked() signals of a set of radio buttons to a station name display so that the display reflects the station chosen by the user, even though the user didn't interact directly with the display.

GUI Events

- Mouse Events
  - Mouse left/right click
  - Mouse double click
  - Mouse enter
  - Mouse Exit
- Key Events
  - Key Press
  - Key Release
  - Key short cuts (Ctrl+c, Ctrl+x etc..)

GUI Events

- Other events
  - Mouse wheel rotate (up, down)
  - Focus In
  - Focus Out
  - Close

Calculation of Features

- Mouse clicks (*lc, rc, and dc)*
  - Average number of left, right and double mouse clicks per user session.
- Mouse enter and exit (*en, ex*)
  - Average number of mouse entrance and exits into the window
- Keys pressed *(kp)*
  - Average number of keys pressed per user session
- Keyboard Shortcuts used (ks*)*
  - Average number of shortcuts used per user session
- Wheel Rotations (*wu, wd)*
  - Average number of up and down wheel rotations

Calculation of Features

- The total of these raw features is 9. Additionally, we calculated the mean 'm' and standard deviation 'sd' for all the raw features above. This gives a total of 18 unique features
- We also calculated the mouse clicks (lc and rc, dc) per 10 minutes of activity for a sliding window starting from the first instance.

Calculation of Features

- If the session starting time t = 31, then we calculate the clicks for 10 minute period following 31 (first period would be [31-41] and second will be [41-51] and so on).
- After finishing this iteration, we slide the window to time t = 32 and calculate the clicks for the following period such that the first period would be the range [32-42], second will be [42 - 52] and so on. We calculate the values for 10 iterations, after that the pattern repeats itself.

Proposed Method

- Sanitization

  -Sanitization is eliminating unnecessary or duplicate (redundant) data

- Feature extraction

  -user behaviour are identified to construct unique feature vectors

Why SVM?

- Text Categorization with Support Vector Machines: Journal of Machine Learning Research (2003) 1265-1287

- SVM for detecting masquerader will works well (Kim, “Efficient Masquerade Detection using SVM based on Common Command Frequency In Sliding Windows”) (Joachims 2001, “A Statistical Learning Model of Text Classification for SVM”)

SVM (Support Vector Machine)

What is SVM

- Developed by **Vapnik** (1995)
- At first, Designed for **Binary Classification**
- Now, Using various field such as bioinformatics, text-classification, artificial Neural Network

key Point : Classifies data by determining a set of support vectors, which are members of the set of training inputs that outlining hyper plane

SVM

- Uses of SVM

  One of the successful uses of SVM algorithms is the task of text categorization into fixed number of predefined categories based on their content

- They are maximal-margin classifiers

- They have known to be highly effective in text classification

Two Class SVM

- It’s reasonable to assume the negative examples (user/self) to be consistent in a certain way, but positive examples (masquerader data) are different since they can belong to any user.

- Since a true masquerader training data is unavailable, other users stand in their shoes.

Benefits of one-class approach

- Practical Advantages:
  - Much less data collection
  - Decentralized management
  - Independent training
  - Faster training and testing
- No need to define a masquerader, but instead detect "impersonators".

Results & Discussion

- We collected data from 8 user's in our lab using the logging tool

- User A (10 sessions), User B (8 Sessions),
- User C (9 sessions), User D (8 sessions),
- User E (6 sessions), User F (7 sessions),
- User G (6 sessions), User H (8 sessions).

Results for Two class SVM

No.of features	Detection Rate	False positive Rate
8	85.57%	2.93%
9	85.57%	2.93%
10	85.57%	2.93%
16	85.9%	3.42%

One-class SVM Results

- For each target user, we train the SVM using only their legitimate training sessions.
- For eg., consider the training and testing strategy for User D.
- The User D is trained as positive.
- Four positive sessions from User D are used for training and the remaining 4 positive sessions are tested with each other user's sessions as negative.

Result of One-class SVM for User D

User	Hits(%)	FPR(%)
User E	97.24	2.75
User F	95.00	5.00
User G	94.87	5.12
User H	92.42	7.80
Average	94.88	5.16

Results & Discussion

- We have achieved comparatively, high detection rates with a very low false positive rate for less number of features(8), compared to previous approaches in the literature.
- Our experimentation also shows that One-class SVM performs better when we have only the users sessions.

Limitations

- Scalability
- Limited data set with limited users
- Execution time of the method.

Conclusion

- We have designed new frame work for capturing GUI events
- We have collected, sanitized and extracted the unique events from the data
- We have constructed feature vectors from these events and used SVM (SVM Light) to first train and then test the users by both One-class and two class SVMs.
- Our results demonstrate One-class SVM is better than Two-class not only the detection rate but also in terms of training and testing times.

Future Work

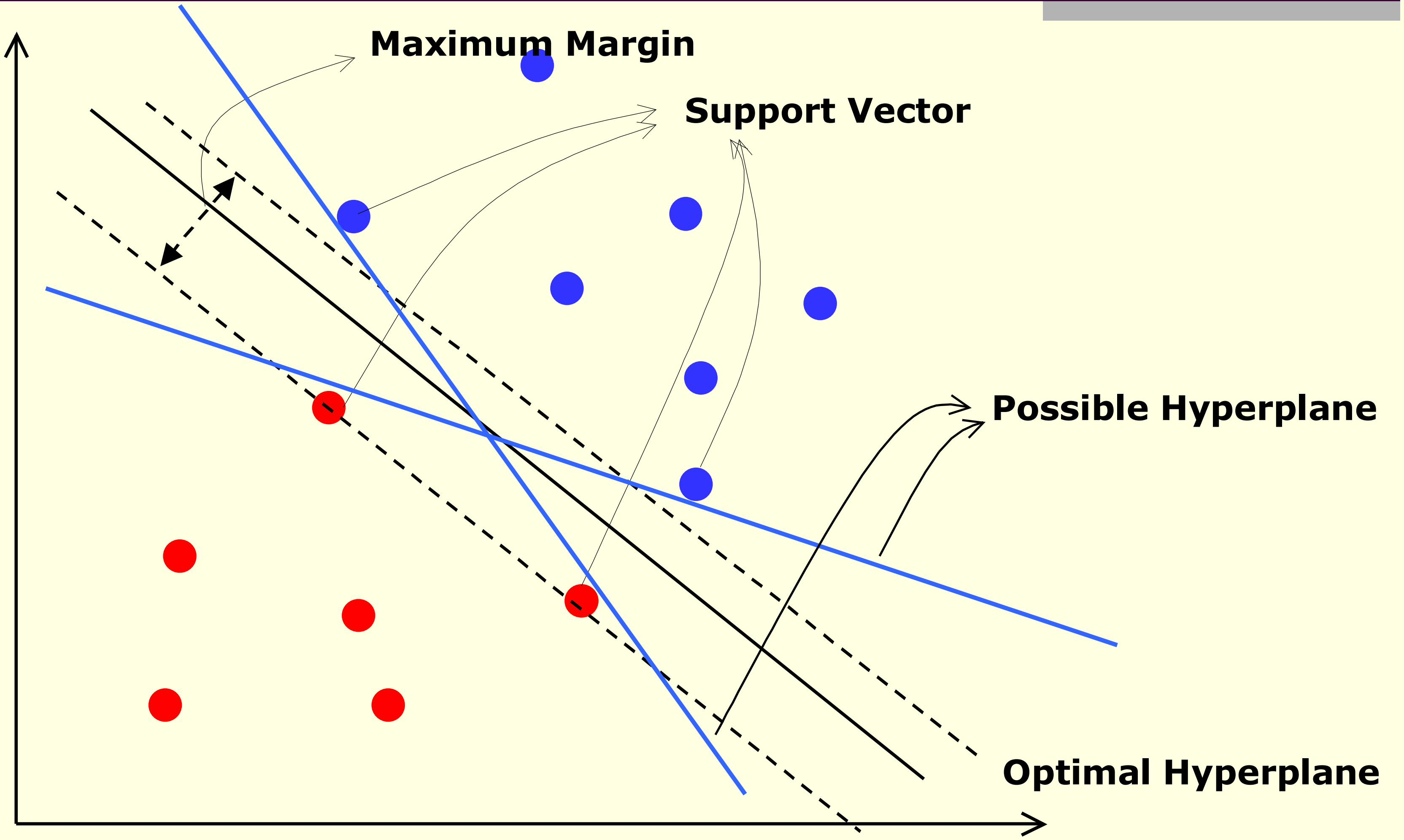
- We plan to capture the events at X-server level .
- We would also like to test our system with more number of users.
- We would like to see this problem in a layered approach by capturing the events across the system (command line, logs, X-server and GUI usage)

References

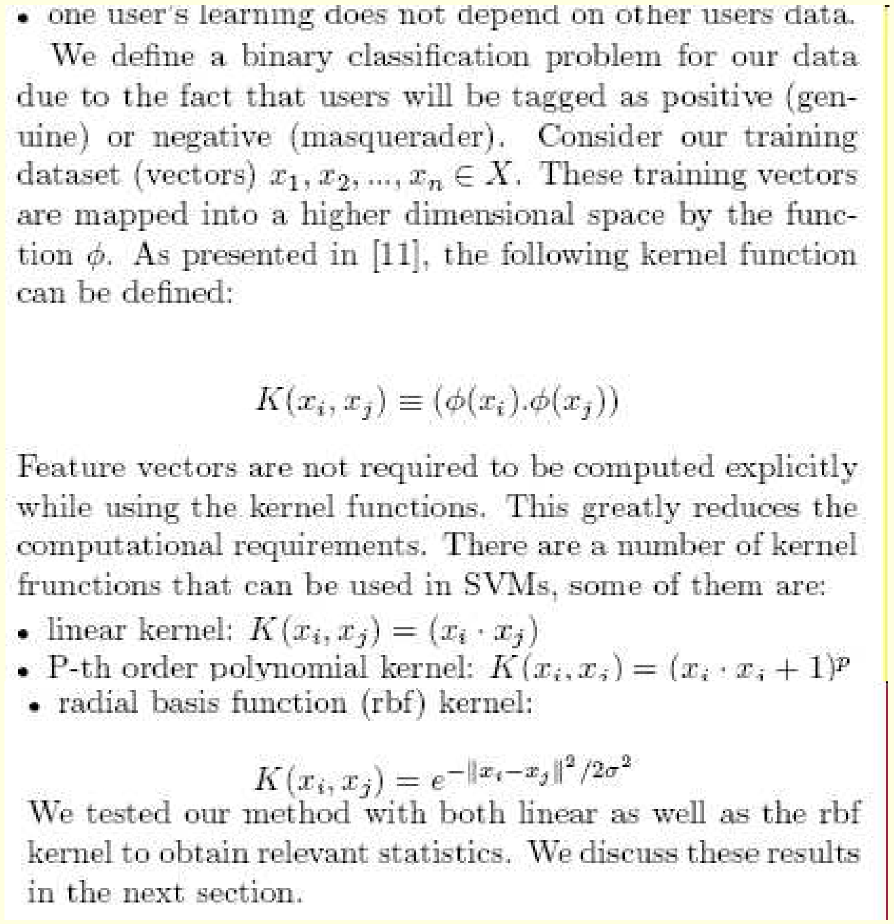
- Ashish Garg, Ragini Rahalkar,Shambu Upadhyaya, Kevin Kwait \Pro¯ling Usersin GUI Based Systems for Masquerade Detection,", in Proceedings of 7th Annual IEEE Information Assurance Workshop (IAW 2006), United States Military Academy, West Point, New York, 21-23 June, 2006.
- Ling Li,Manikopoulos, Windows NT One-Class Masquerade Detection,", in Proceedings of 2004 IEEE,Information Assurance Workshop (IAW 2004), United States Military Academy, West Point, New York, June 2004.
- T. Lane and C. E. Brodley, \An Application of Machine Learning to Anomaly Detection", In Proceedings of Twentieth National Information Systems Security Conference, vol. 1, (Gaithersburgh, MD), pp. 366-380, 1997. The National Institute of Standards and Technology and the National Computer Security Center.
- M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, and M. T. and Y. Vardi, Computer Intrusion: Detecting Masquerades", In Statistical Science, vol. 16, pp. 58-74, 2001. R. A. Maxion and T. N. Townsend, \Masquerade Detection Using Truncated Command Lines", In Proceedings of International Conference on Dependable Systems and Networks (DSN '02), pp. 219-228, 2002
- K. Wang and S. J. Stolfo, \One Class Training for Masquerade Detection", In ICDM Workshop on Data Mining for Computer Security (DMSEC 03), 2003.

Thank you

What is SVM?



Classification by SVM



- one user's learning does not depend on other users data.

We define a binary classification problem for our data due to the fact that users will be tagged as positive (genuine) or negative (masquerader). Consider our training dataset (vectors) $x_1, x_2, ..., x_n \in X$. These training vectors are mapped into a higher dimensional space by the function ϕ. As presented in [11], the following kernel function can be defined:

$$K(x_i, x_j) \equiv (\phi(x_i).\phi(x_j))$$

Feature vectors are not required to be computed explicitly while using the kernel functions. This greatly reduces the computational requirements. There are a number of kernel frunctions that can be used in SVMs, some of them are:

- linear kernel: $K(x_i, x_j) = (x_i \cdot x_j)$
- P-th order polynomial kernel: $K(x_i, x_j) = (x_i \cdot x_j + 1)^p$
- radial basis function (rbf) kernel:

$$K(x_i, x_j) = e^{-\|x_i - x_j\|^2/2\sigma^2}$$

We tested our method with both linear as well as the rbf kernel to obtain relevant statistics. We discuss these results in the next section.