Spring 2016

Dueling-HMM Analysis on Masquerade Detection

Peter Chou
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Information Security Commons](#)

Recommended Citation

Chou, Peter, "Dueling-HMM Analysis on Masquerade Detection" (2016). *Master's Projects*. 464.
DOI: <https://doi.org/10.31979/etd.497c-u3tp>
https://scholarworks.sjsu.edu/etd_projects/464

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Dueling-HMM Analysis on Masquerade Detection

A Project

Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Peter Chou

May 2016

© 2016

Peter Chou

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Project Titled

Dueling-HMM Analysis on Masquerade Detection

by

Peter Chou

APPROVED FOR THE DEPARTMENTS OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2016

Thomas H. Austin Department of Computer Science

Mark Stamp Department of Computer Science

Robert K. Chun Department of Computer Science

ABSTRACT

Dueling-HMM Analysis on Masquerade Detection

by Peter Chou

Masquerade detection is the ability to detect attackers known as masqueraders that intrude on another user's system and pose as legitimate users. Once a masquerader obtains access to a user's system, the masquerader has free reign over whatever data is on that system. In this research, we focus on masquerade detection and user classification using the following two different approaches: the heavy hitter approach and 2 different approaches based on hidden Markov models (HMMs), the dueling-HMM and threshold-HMM strategies.

The heavy hitter approach computes the frequent elements seen in the training data sequence and test data sequence and computes the distance to see whether the test data sequence is masqueraded or not. The results show very misleading classifications, suggesting that the approach is not viable for masquerade detection.

A hidden Markov model is a tool for representing probability distributions over sequences of observations [9]. Previous research has shown that using a threshold-based hidden Markov model (HMM) approach is successful in a variety of categories: malware detection, intrusion detection, pattern recognition, etc. We have verified that using a threshold-based HMM approach produces high accuracy with low amounts of a false positives. Using the dueling-HMM approach, which utilizes multiple training HMMs, we obtain an overall accuracy of 81.96%. With the introduction of the bias in the dueling-HMM approach, we produce similar results to the results obtained in the threshold-based HMM approach, where we see many non-masqueraded data detected, while many masqueraded data avoid detection, yet still result in an high overall accuracy.

ACKNOWLEDGMENTS

I want to thank Professor Thomas H. Austin for the great advise throughout this project. I would also like to thank my committee members Professor Mark Stamp and Professor Robert K. Chun for taking time out of their busy schedule to attend my defense and review my report.

TABLE OF CONTENTS

CHAPTER

1	Introduction	1
2	Masquerade Detection Overview	3
2.1	Masquerade Detection	3
2.2	Schonau Data Set	4
3	Heavy Hitter	8
3.1	Heavy Hitter Background Information	8
3.2	Heavy Hitter Implementation	9
3.2.1	Space-saving Algorithm	9
3.2.2	Levenshtein Distance	10
3.3	Heavy Hitter Testing	12
3.4	Heavy Hitter Results	14
3.4.1	Space-Saving Result	14
3.4.2	Levenshtein Distance Results	17
4	Hidden Markov Models	20
4.1	Hidden Markov Models	20
4.1.1	K-Means Learning	21
4.1.2	Smoothing	22
4.1.3	Estimator	22
5	Threshold-Based HMM Approach	23
5.1	Threshold-based HMM Implementation	23

5.1.1	Threshold-Based General Implementation Scheme	23
5.1.2	Threshold-Based HMM Testing	24
5.2	Threshold-base HMM results	25
6	Dueling-HMM Approach	30
6.1	Dueling-HMM Implementation	30
6.1.1	Dueling-HMM: Detecting Masqueraded Data Implementation . . .	30
6.2	Dueling-HMM: User Classification Implementation	32
6.3	Dueling-HMM Testing	34
6.4	Dueling-HMM Results	36
6.4.1	Dueling-HMM Masquerade Detection Results	36
6.4.2	Dueling-HMM Bias Results	36
6.4.3	Dueling-HMM User Classification Results	37
6.4.4	Dueling-HMM User Classification Confidence Factor Results . . .	38
7	Conclusion	42
APPENDIX		
A	Heavy Hitter Space Saving Results	48
B	Threshold-Based HMM Threshold Results	52
C	Dueling-HMM Bias Results	54

LIST OF TABLES

1	HH - Top 3 Results	15
2	HH - Top 5 Results	16
3	HH - Top 10 Results	17
4	Levenshtein Distance Results	18
5	Benign-Only Threshold-Based HMM Results	26
6	Benign-Only Threshold-Based HMM Threshold Results	29
7	Dueling-HMM Results	37
8	Dueling-HMM Bias Result	38
9	User Classification Results	40
10	User Classification Confidence Factor Results	41

LIST OF FIGURES

1	Schonlau Data Set Structure [5]	5
2	Schonlau Solution Set Structure [5]	5
3	<i>Stream-Summary</i> updates with $m = 2$ [1]	11
4	Counter-Based Flow Diagram	12
5	Markov State Graph [5]	21
6	General Implementation Scheme Flow	24
7	Threshold ROC Plot	27
8	Schonlau ROC Plot [27]	28
9	Dueling-HMM Masquerade Detection Flow	32
10	Dueling-HMM User Classification Flow	33
11	Dueling-HMM Bias ROC Plot	39

CHAPTER 1

Introduction

A masquerader is an attacker that appears to be a normal user with valid authority and privileges [20]. An attack from a masquerader is considered one of the most damaging attacks in the field of computer security [20] [21] [22]. In order to accurately detect a masquerader, we must be able to distinguish between users by observing the behaviors or patterns in which a user issues a sequence of commands collected from his or her system.

The masquerade detection approach and user classification approach we use in this research are the heavy hitter approach [1] and the dueling-HMM approach [23]. The heavy hitter approach has been applied to mostly intrusion detections related to networks [24], so we decided to see if the heavy hitter approach would be applicable to masquerade detection. The heavy hitter approach utilizes the frequency in which a user has issued a command. By computing the heavy hitters, aka most frequent elements, we get a list of the top k elements from the training data sequence for a particular user. We associate this list of top k elements to represent that particular user. We then compute the top k elements for the 100 test data sequences associated to each user.

With the top k elements computed for the 100 test data sequences, we then use these computed top k frequent elements to match the top k elements in the training sequence. The results were unsatisfactory in that over 90% of the test cases would return less than 50% match. Adding onto the heavy hitter approach, we use the Levenshtein distance approach to compute the distance between the top k elements from the test sequence to the top k elements from the training sequence. Further details about the Levenshtein distance approach is in Chapter 3. Our test results show that many of the test sequences would be misclassified,

though we do see an overall increase in accuracy by 30%. The results we obtained using the heavy hitter approach was unsatisfactory, so we proceeded to experimenting with the following two HMM-based approaches: threshold-based HMM and dueling-HMM.

Previous research has shown that using a threshold-based hidden Markov model (HMM) approach is successful in a variety of categories: malware detection [28], intrusion detection [29], pattern recognition [9], etc. In this research, we verify that a threshold-based HMM approach was able to capture most of the masquerade data.

The dueling-HMM approach utilizes multiple training HMMs to detect masqueraders and classify users. For masquerade detection, each user would have two training HMMs, one benign-only HMM and one masquerade-only HMM. We use an Estimator from the JAHMM utils project [33] to compute the distance between the test data sequences and the two training HMMs. We achieve an overall accuracy of 81% using the dueling-HMM approach. To see if we could improve those results, we introduce a bias. With the introduction of a bias, we are able to achieve 95% accuracy. For user classification, we use only the benign-only training HMMs. So in total, we have 50 training HMMs, each representing a user, and use only the non-masqueraded test sequences to see if this approach can correctly classify which test sequence belongs to which user.

CHAPTER 2

Masquerade Detection Overview

2.1 Masquerade Detection

As the word masquerade implies, we are trying to detect whether a user is impersonating another user. In sci-fi movies, we see spies wear masks of certain individuals to fool facial recognition programs, or use the certain individuals handprint to fool hand recognition softwares. Impersonations come in many different forms: physical movements, gestures, handwriting, voice, facial make-overs, etc.

Impersonation in the cyber world can occur when intruders successfully hack into another user's account and use his or her account to do whatever he or she wants. When an email account gets hacked, we normally see that hacked account start spamming users in that email accounts contact list. This spamming behavior is quite a clear indication that the original account's owner is being impersonated and used for some kind of attack.

Let's consider the following scenario. We have an actual user, Bob, who is an engineer who runs many Linux commands all day on his work system. An intruder, Joker, wants to hack into Bob's system and run all sorts of malware on Bob's system without being detected. Joker knows Bob runs thousands of commands a day, and tries to impersonate Bob by running similar commands that Bob would use. If Joker manages to replicate Bob's behavior through the Linux commands Bob uses, then Joker has effectively been able to masquerade as Bob. When that happens, we would have an extremely tough time trying to distinguish whether Bob was the one who ran the different malware programs or whether some intruder masqueraded as him and ran those malware softwares.

2.2 Schonlau Data Set

For this project, we use the Schonlau data set [4] to see how accurately the heavy hitter approach and dueling-HMM approach would be able to classify users and perform masquerade detection. We use the Schonlau data set because it has become the standard data set for masquerade detection studies [5]. Figure 2 illustrates the structure of the Schonlau data set.

The Schonlau data set consists of 50 text files, and each file corresponds to one user. Each file consists of 15,000 commands that were collected using the UNIX audit tool. The first 5,000 commands capture the actual users' behavior, which is intended to serve as our training data. We consider this data as non-masqueraded data. The following 10,000 commands include both non-masqueraded and masqueraded blocks of commands, which is intended to serve as our test data. Each block consists of 100 lines of commands.

At any given block after the initial 5000 commands a masquerade starts with a probability of 1%. If the previous block was a masquerade, the next block will also be a masquerade with a probability of 80%. About 5% of the test data contain masquerades" [4].

The Schonlau website already contains a text file that illustrates which blocks are masqueraded data for each user. The solutions file is in the format where each column corresponds to a user, and each row corresponds to a test block. The value associated with each test block is either 0 or 1. A zero implies that the block of data is non-masqueraded, and a one implies that the block of data is masqueraded.

Through the survey article of Bertacchini and Fierens [2] and the article of Erbacher et al. [7], we compiled a list of general approaches done on the Schonlau data set to detect masqueraded data.

- **Information-theoretic** - This approach is based off the compression-based approach

Figure 1: Schonlau Data Set Structure [5]

Figure 2: Schonlau Solution Set Structure [5]

that Schonlau et al. proposed called the *compression* method [2]. The compress method utilizes the **compress** UNIX command to compress data that either consists of the same user or mixed users.

- **Text mining** - This approach is based off the *sequitur* algorithm [13], which extracts hierarchical structures from a string by constructing a context-free grammar [13].
- **Hidden Markov Model (HMM)** - An HMM is defined as a doubly stochastic process with an underlying process that is hidden [18]. The hidden stochastic process

cannot be directly observed. It can only be observed through a process that produces a sequence of observed symbols [18]. Past researchers have used the hidden Markov model to construct user profiles and used it to score testing blocks by means of some voting mechanism and a threshold value to determine the legitimacy of that testing block.

- **Naive Bayes** - This approach is known for its inherent robustness to noise and its fast learning curve [2]. It generally uses the "bag of words" model [16], which profiles documents based on word frequencies and ignores the sequence information.
- **Sequences Matching** - This approach is commonly used to quantify the similarity between two genetic sequences [2]. Schonlau's use of this approach modeled after the work of T. Lane and C. E. Brodley [17]. The *sequence match* approach measures the ten most recent commands from the test data and a user's profile by counting the number of matches [17].
- **Support vector machine (SVM)** - The SVM approach utilizes a a set of machine learning algorithms used for classification and regression [2].
- **Uniqueness** - As the name implies, this approach is based on the idea that commands not previously seen in the training data may indicate an attempted masquerade [14]. In essence, the less frequent a command is used by users, the more that command is seen as a masquerade.
- **Hybrid Multi-Step Markov** - This approach utilizes a multi-step Markov chain and occasionally an independence model [15]. When a case such as the Markov model not being usable when the test data contains many unobserved commands, the independence model takes action and estimates the probabilities from a contingency table of users versus commands.

- **Bayes one-step Markov** - The Bayes one-step Markov goes further than the *uniqueness* approach that only utilizes the command frequencies [14]. It is based on observing the one-step transition probabilities from one command to the next command.

To our knowledge, neither the heavy hitters approach nor the dueling-HMM approach have previously been studied in this domain. The following chapters discuss some background information, implementation, tests, and results for the different approaches.

CHAPTER 3

Heavy Hitter

This chapter will discuss the implementation methods for the heavy hitter approach.

3.1 Heavy Hitter Background Information

"Heavy hitters" is the term used to refer to the most frequent (popular) elements in a data stream while efficiently utilizing the space within the environment. Large sources of data, such as streams of network packets, are usually best modeled as data streams. Data streams are usually a large amount of data that is impractical and undesirable to store and impractical and undesirable to process data exactly.

Heavy hitter is also referred to as the "Frequent Items Problem, where given a stream of N items, find those that occur most frequently" [8]. The goal of finding the most frequent items from a stream of data can help us in many different analysis cases: data mining [31], network data analysis [24], bandwidth [32], etc.

Using the basis of finding frequent elements in a data stream, we use a heavy hitter algorithm on the Schonlau data set to compute the heavy hitters for each user. These heavy hitters are the top k elements, or most frequently seen commands, in the data sequence. We associate heavy hitters computed from the training set as that particular user's behavior and see if we can determine whether we can detect masqueraded behavior from the test data.

There are a few popular heavy hitter algorithms, and they can be split into two categories: counter-based algorithms and sketch algorithms [8]. Counter-based algorithms solve the problems in arrival-only models. Arrival only models are streams where data is being accumulated. Counter-based algorithms keep an individual counter for each element in

a monitored set of elements. Sketch algorithms handle problems in arrival-and-departure models where streams contain fluctuating data. Unlike the counter-based algorithms, sketch algorithms do not have a monitored set of elements, but provide, with less stringent guarantees, frequency estimation for all elements by using bit-maps of counters [1].

The Schonlau data set is a static data set and does not contain any fluctuating data, so for this project, we use the counter-based algorithm that utilizes space saving.

3.2 Heavy Hitter Implementation

Through the use of the heavy hitter algorithm, we experiment to see if we can detect masqueraded data by comparing and computing the top k elements of a user’s training data sequence with the top k elements from the user’s test data sequences. Two experiments were done using this approach, the *space-saving* algorithm and Levenshtein distance. We implement the *space-saving* algorithm to compute the top k elements from the training data sequence and the test data sequences in Java.

3.2.1 Space-saving Algorithm

The *space-saving* algorithm we use in this project follows the *Stream-Summary* data structure approach discussed in the article of Metwally et al. [1]. The *Stream-Summary* consists of buckets that contain a linked list of elements that have the same counter value. In our implementation, our *Stream-Summary* is represented as a LinkedHashMap with the counter value as the key, and the linked list of elements as the value.

Algorithm 1 displays the general steps of the *space-saving* algorithm used in this project, which is based off of Metwally et al.’s algorithm [1]. What the *space-saving* algorithm does is monitor the fast-growing elements in the data provided. This saves the space considerably for large or would be infinite amounts of data. The parameter m is an integer that specifies

how many elements to monitor.

Algorithm 1 Space-Saving (int m, String fileName)

```
begin
while !EOF
    create an Element, e, with the value read in
    if e, is monitored
        // Checks the monitored list of elements
        // and updates the count of the existing element
        // in the list
        let count_i be the counter of e
        Increment-Counter(count_i)
    else
        // Replacement Step
        if size of monitored list < m
            Add element to the monitored list
        else
            Get the element with the least hit, e_m,
            from the monitored list
            Replace e_m with e
            Increment-Counter(count_m)
end
```

Algorithm 2 displays the general steps of the *Increment-Counter* algorithm used in this project. This is where we update the *Stream-Summary* by moving the Counters to the Bucket with the correct Counter value. Figure 3 is a visual representation of how the *Stream-Summary* works.

Figure 4 displays the flow diagram for the counter-based heavy hitter algorithm.

3.2.2 Levenshtein Distance

The Levenshtein distance is a metric for measuring how different two sequences are. The distance between two sequences is defined as the minimum number of edits needed to transform one sequence into the other through the following edit operations: insertion,

Algorithm 2 Increment-Counter (Counter c_i)

```
begin
if Stream-Summary is not Empty
    Remove the  $c_i$  from the bucket it is currently in
    if the bucket_i is empty
        Detach bucket_i from the Stream-Summary
        Delete bucket_i

increase the count of  $c_i$ ,  $count_i$ 
if bucket with new incremented count value exists
    Attach  $c_i$  to the corresponding bucket's list
else
    create a new bucket
    assign bucket_new the value of  $count_i$ 
    Attach  $c_i$  to the new bucket
    Insert new bucket into the Stream-Summary
end
```

Figure 3: *Stream-Summary* updates with $m = 2$ [1]

deletion, or substitution. For example, the Levenshtein distance between two strings, "bite" and "hitting", is 5. The edits needed would be the following:

```
bite hitting (substitution of 'b' with 'h')
hite hitting (insert 't' after 't')
hitte hitting (substitution of 'e' with i)
hitti hitting (insert 'n' at the end)
hittin hitting (insert 'g' at the end)
```

Figure 4: Counter-Based Flow Diagram

The first experiment used only the *space-saving* algorithm to compute the top k elements and then determine whether a test data sequence was masqueraded or not by comparing how many match the top k elements of that user. The second experiment utilized the Levenshtein distance to compute the similarity between the top k elements obtained from the *space-saving* algorithm to determine whether a test sequence is masqueraded or not.

3.3 Heavy Hitter Testing

With the heavy hitter approach, we split the Schonlau data set into two sets of data for each user: a training set and a testing set. The training set contains the 5,000 lines of

Linux commands that were classified as benign and the testing set contains the 10,000 lines of Linux commands following the first 5,000 lines. We run our *space-saving* algorithm that is implemented in Java to compute the top k heavy hitters from the training set. In order to compute the top k heavy hitters in the testing set, we needed to further break the testing set into sequences of 100 lines of Linux commands. We then compute the top k heavy hitters for the 100 test sequences. To determine whether a test sequence is masqueraded or not, we compare the two sets of top k heavy hitters and see how many from the test sequence match the top k heavy hitters obtained from the training sequence.

The results from just comparing the union of the two sets of heavy hitters found from the test sequences and training sequence produces a very low accuracy as seen in the Table 1, Table 2, and Table 3 in Section 3.4. Building up on the heavy hitter approach, we explore using the Levenshtein distance on the heavy hitters we computed to see if we can improve the overall accuracy.

The Levenshtein distance in the example provided in Section 3.2.2 takes two strings and computes the distance using the characters in each string. Instead of computing the distances between two strings, we are using the Levenshtein distance to compute the distance between two array of strings. Let's look at an example below of two arrays of strings, testing array and training array. The testing array contains: sh, ls, grep, mkdir, top. The training array contains: sh, grep, ps, mkdir.

```
TEArray = Testing Array
TRArray = Training Array
```

Edit 1:

```
TEArray = [sh, ls, grep, mkdir, top]
TRArray = [sh, grep, ps, mkdir] (substitution of 'ls' to 'grep')
```

Edit 2:

```
TEArray = [sh, grep, grep, mkdir, top]
```

```
TRArray = [sh, grep, ps, mkdir] (substitution of 'grep' to 'ps')
```

Edit 3:

```
TEArray = [sh, grep, ps, mkdir, top]
```

```
TRArray = [sh, grep, ps, mkdir] (deletion of 'top')
```

Following the example above, we require 3 edits; therefore, the distance is 3. Using the Levenshtein distance, we improve the overall accuracy to about 43.5% shown in Table 4 in Section 3.4.

3.4 Heavy Hitter Results

We use the heavy hitter approach as is and try to improve upon the heavy hitter approach using the Levenshtein distance. In the first experiment, we use the *space-saving* algorithm to compute the top k elements and compare the top k elements of the test set and top k elements of the training set and see if they match. The second experiment follows the same procedure as the first experiment; however, instead of just comparing how many of the top k elements matched, we use the Levenshtein distance algorithm to compute the distance between the two arrays of top k heavy hitters from the training set and the testing sets to generate our results.

3.4.1 Space-Saving Result

Table 1, Table 2, and Table 3 show the results of the heavy hitter *space-saving* approach when we set the the algorithm to monitor the Top 3, 5, and 10 elements respectively. The heavy hitter *space-saving* results table shown in this section displays the results of using a threshold value that determines if the test was not masqueraded if the value of top k elements in the testing data matched greater than 50% of the top k elements seen in the training data.

Table 1, which uses top 3 elements, produces the best overall accuracy of 19.34% out of

the three different variations. It has the least amount of false alarms, False Positives (FP), and also detected the least amount of masqueraded data, True Positive (TP); however, detecting a higher amount of actual user behavior, True Negatives (TN). We see that only the top 3 elements results misclassified masqueraded data, False Negatives (FN).

Table 1: HH - Top 3 Results

User	Accuracy	TP	FP	TN	FN	User	Accuracy	TP	FP	TN	FN
User 1	30%	0	70	30	0	User 2	32%	3	68	29	0
User 3	39%	11	61	28	0	User 4	3%	2	97	1	0
User 5	0%	0	100	0	0	User 6	35%	0	65	35	0
User 7	13%	13	87	0	0	User 8	0%	0	100	0	0
User 9	37%	24	63	13	0	User 10	19%	13	81	6	0
User 11	2%	0	98	2	0	User 12	6%	6	94	0	0
User 13	5%	0	95	5	0	User 14	0%	0	100	0	0
User 15	7%	6	93	1	0	User 16	11%	10	89	1	0
User 17	36%	0	64	36	0	User 18	38%	6	62	32	0
User 19	53%	0	47	53	0	User 20	0%	0	100	0	0
User 21	8%	0	92	8	0	User 22	0%	0	100	0	0
User 23	1%	1	99	0	0	User 24	25%	21	75	4	0
User 25	9%	9	91	0	0	User 26	16%	11	82	5	2
User 27	1%	0	99	1	0	User 28	17%	3	83	14	0
User 29	3%	1	97	2	0	User 30	37%	3	63	34	0
User 31	1%	0	99	1	0	User 32	34%	0	66	34	0
User 33	19%	0	81	19	0	User 34	21%	12	79	9	0
User 35	75%	1	25	74	0	User 36	46%	6	54	40	0
User 37	26%	2	73	24	0	User 38	29%	7	69	22	2
User 39	11%	0	89	11	0	User 40	6%	0	94	6	0
User 41	8%	3	92	5	0	User 42	25%	20	75	5	0
User 43	35%	16	65	19	0	User 44	6%	6	94	0	0
User 45	30%	5	70	25	0	User 46	42%	4	58	38	0
User 47	39%	0	61	39	0	User 48	19%	2	81	17	0
User 49	4%	0	96	4	0	User 50	8%	0	92	8	0
Total	19.34%	227	4029	740	4						

Summary: This table shows the results when we monitor the Top 3 Elements and set a threshold value of 2 to determine whether a test is masqueraded or not. The overall accuracy is 19.34%.

Table 2, which uses top 5 elements, obtained the second best overall accuracy of 18.7%. We see slightly higher amounts of False Positives than when we use the top 3 elements, which

resulted in slightly lower overall accuracy.

Table 2: HH - Top 5 Results

User	Accuracy	TP	FP	TN	FN	User	Accuracy	TP	FP	TN	FN
User 1	23%	0	77	23	0	User 2	51%	3	49	48	0
User 3	61%	11	39	50	0	User 4	3%	2	97	1	0
User 5	4%	0	96	4	0	User 6	15%	0	85	15	0
User 7	15%	13	85	2	0	User 8	0%	0	100	0	0
User 9	30%	24	70	6	0	User 10	27%	13	73	14	0
User 11	9%	0	91	9	0	User 12	6%	6	94	0	0
User 13	2%	0	98	2	0	User 14	0%	0	100	0	0
User 15	9%	6	91	3	0	User 16	11%	10	89	1	0
User 17	46%	0	54	46	0	User 18	12%	6	88	6	0
User 19	42%	0	58	42	0	User 20	0%	0	100	0	0
User 21	48%	0	52	48	0	User 22	2%	0	98	2	0
User 23	1%	1	99	0	0	User 24	22%	21	78	1	0
User 25	9%	9	91	0	0	User 26	15%	13	85	2	0
User 27	0%	0	100	0	0	User 28	15%	3	85	12	0
User 29	1%	1	92	0	0	User 30	34%	3	66	31	0
User 31	2%	0	98	2	0	User 32	36%	0	64	36	0
User 33	38%	0	62	38	0	User 34	15%	12	85	3	0
User 35	18%	1	82	17	0	User 36	35%	6	65	29	0
User 37	11%	2	89	9	0	User 38	22%	9	78	13	0
User 39	14%	0	86	14	0	User 40	5%	0	95	5	0
User 41	8%	3	92	5	0	User 42	22%	20	78	2	0
User 43	34%	16	66	18	0	User 44	13%	6	87	7	0
User 45	10%	5	90	5	0	User 46	47%	4	53	43	0
User 47	13%	0	87	13	0	User 48	27%	2	73	25	0
User 49	4%	0	96	4	0	User 50	5%	0	95	5	0
Total	17.84%	231	4108	661	0						

Summary: This table shows the results when we monitor the Top 5 Elements and set a threshold value of 3 to determine whether a test is masqueraded or not. The overall accuracy is 17.84%.

Lastly, Table 3, which uses top 10 elements, obtained the lowest overall accuracy of 2.96%. We used the threshold greater than 60%, so if a test block contains at least 6 out of 10 matches, we classify it as non-masqueraded. If we lowered the threshold value to 5 or less, the overall accuracy would of course improve; however, if masqueraders only need to match less than 50% of that user’s behavior, it would be all too easy for someone to masquerade

as a user. Appendix A contains the table results for when we use top 10 elements with a threshold of 2, 3, 4, and 5.

Table 3: HH - Top 10 Results

User	Accuracy	TP	FP	TN	FN	User	Accuracy	TP	FP	TN	FN
User 1	8%	0	92	8	0	User 2	40%	3	60	37	0
User 3	30%	11	70	19	0	User 4	2%	2	98	0	0
User 5	4%	0	96	4	0	User 6	48%	0	52	48	0
User 7	13%	13	87	0	0	User 8	0%	0	100	0	0
User 9	26%	24	74	2	0	User 10	21%	13	79	8	0
User 11	5%	0	95	5	0	User 12	6%	6	94	0	0
User 13	0%	0	100	0	0	User 14	0%	0	100	0	0
User 15	6%	6	94	0	0	User 16	10%	10	90	0	0
User 17	49%	0	51	49	0	User 18	6%	6	94	0	0
User 19	47%	0	53	47	0	User 20	0%	0	100	0	0
User 21	34%	0	66	34	0	User 22	5%	0	95	5	0
User 23	1%	1	99	0	0	User 24	21%	21	79	0	0
User 25	9%	9	91	0	0	User 26	13%	13	87	0	0
User 27	0%	0	100	0	0	User 28	3%	3	97	0	0
User 29	1%	1	99	0	0	User 30	3%	3	97	0	0
User 31	1%	0	99	1	0	User 32	26%	0	74	26	0
User 33	31%	0	69	31	0	User 34	13%	12	87	1	0
User 35	5%	1	95	4	0	User 36	15%	6	85	9	0
User 37	6%	2	94	4	0	User 38	24%	9	76	15	0
User 39	3%	0	97	3	0	User 40	2%	0	98	2	0
User 41	3%	3	97	0	0	User 42	20%	20	80	0	0
User 43	17%	16	83	1	0	User 44	7%	6	93	1	0
User 45	5%	5	95	0	0	User 46	42%	4	58	38	0
User 47	8%	0	92	8	0	User 48	7%	2	93	5	0
User 49	1%	0	99	1	0	User 50	1%	0	99	1	0
Total	12.96%	231	4352	417	0						

Summary: This table shows the results when we monitor the Top 10 Elements and set a threshold value of 6 to determine whether a test is masqueraded or not. The overall accuracy is 12.96%.

3.4.2 Levenshtein Distance Results

A few different variation of the Levenshtein distance algorithms were experimented with. However, none provided desirable results. Although most algorithms would be able

to classify most of the masqueraded data, but that is because at least around 50% of the results would return a high distance value, which represents that many edits were required.

Table 4: Levenshtein Distance Results

User	Accuracy	TP	FP	TN	FN	User	Accuracy	TP	FP	TN	FN
User 1	40%	0	60	40	0	User 2	93%	3	7	90	0
User 3	23%	11	77	12	0	User 4	50%	2	50	48	0
User 5	71%	0	29	71	0	User 6	88%	0	12	88	0
User 7	45%	13	55	32	0	User 8	12%	0	88	12	0
User 9	52%	22	46	30	2	User 10	36%	13	64	23	0
User 11	17%	0	83	17	0	User 12	11%	6	89	5	0
User 13	32%	0	68	32	0	User 14	0%	0	100	0	0
User 15	20%	5	79	15	1	User 16	12%	7	85	5	3
User 17	73%	0	27	73	0	User 18	25%	6	75	19	0
User 19	80%	0	20	80	0	User 20	1%	0	99	1	0
User 21	59%	0	41	59	0	User 22	13%	0	87	13	0
User 23	3%	1	97	2	0	User 24	64%	21	36	43	0
User 25	32%	9	68	23	0	User 26	47%	10	50	37	3
User 27	3%	0	97	3	0	User 28	31%	3	69	28	0
User 29	17%	1	83	16	0	User 30	100%	3	0	97	0
User 31	14%	0	86	14	0	User 32	44%	0	56	44	0
User 33	61%	0	39	61	0	User 34	37%	8	59	29	4
User 35	23%	0	76	23	1	User 36	48%	6	52	42	0
User 37	34%	2	65	32	0	User 38	36%	8	63	28	1
User 39	22%	0	78	22	0	User 40	23%	0	77	23	0
User 41	6%	3	94	3	0	User 42	25%	11	66	14	9
User 43	38%	16	62	22	0	User 44	11%	6	89	5	0
User 45	59%	5	41	54	0	User 46	73%	4	27	69	0
User 47	30%	0	70	30	0	User 48	47%	2	53	45	0
User 49	3%	0	97	3	0	User 50	15%	0	85	15	0
Total	35.98%	207	3177	1592	24						

Summary: The Levenshtein distance provides an increase in the overall accuracy. It is able to detect more True Negatives; however, the trade-off is missing more masqueraded data.

In Table 4, the overall accuracy in detecting masqueraded data is below 50% for around 90% of the users. User 14 did not contain any masqueraded data, yet the tests all resulted in being classified as masqueraded. Other cases where the number of masqueraded tests amounted to around 70% or above, would generally catch all, if not most, of the masqueraded

test sequences; however, those user's results show a high percentage of false alarms.

CHAPTER 4

Hidden Markov Models

This chapter will go over some basic information about hidden Markov models. More in-depth details can be found in Stamp's article [5].

4.1 Hidden Markov Models

A hidden Markov model is a tool for representing probability distributions over sequences of observations [9]. There are two properties that define the hidden Markov model:

- First, the model assumes that an observation at a particular time t was generated by some process whose state S_t is hidden from the observer.
- Second, the model assumes that the state of this hidden process satisfies the Markov property.

The Markov property states that given the value of S_{t-1} , the current state S_t is independent of all the states prior to $t - 1$ [9]. In simpler terms, we can consider the hidden Markov model as a state graph. In Figure 5, the nodes are the states observed at a particular time, and the edges show the probability of going to another state. The edges seen in the graph form the transition probability matrix A .

Although we cannot directly see what exactly the state belongs to, we can observe the behavior of the state. Using the Schonlau data set as an example, we can observe the sequence of commands a particular user issues and the frequency of using that command to determine the observation sequence matrix B . This observation sequence matrix allows us to contain the probabilities of all the possible observations within a state. One last matrix is

Figure 5: Markov State Graph [5]

needed to kick start the Hidden Markov Model, which is the initial state distribution matrix π . The three matrices, A , B , and π define an HMM.

In this project, we decide to use the HMM with K-means learning and compute the distance using the Estimator, which utilizes the Java HMM (JAHMM) library [10].

4.1.1 K-Means Learning

The K-means learning uses a set of data points where each data point is a vector x of n numbers. A vector x may contain values that we want to try and classify. For example, if we want to prioritize a task, we might have:

$$x_i = (\text{difficulty of task } i, \text{severity of task } i)$$

The K in K-means represents the number of clusters. We want to determine a set of K clusters for our set of data points. We assume there is a distance function $d(x_i, x_j)$ that is defined for all pairs of data points. Each data point will belong to exactly one cluster. In order to determine which cluster that data point belongs in, centroids need to be computed. These centroids determine the clusters, and we compute the distance between a data point and a centroid to determine which cluster the data point belongs to.

4.1.2 Smoothing

Smoothing is a mathematical technique that removes the excess data variability while maintaining a correct appraisal [25]. K-means learning gives us an HMM that would compute probabilities on the data read in from the observation sequences; however, it will not cover the probability distribution of unseen sequences. By using the smoothing process, we smooth the HMM created from K-means learning so that all sequences can occur with some probability.

4.1.3 Estimator

The Estimator is a Java jar file that computes the probability of the test data sequence to a training HMM. The code utilizes the JAHMM library, by using the library's observation sequence readers. The Estimator reads in a test data sequence, then computes for each command in the test data sequence the likelihood in which it is seen in the training HMM. This number will be very small and so a natural log of that value is used to produce a probability. We then use this probability to determine the optimal threshold and bias in the threshold-based HMM approach and dueling-HMM approach.

In Chapter 5 and Chapter 6, we will discuss more details about the threshold-based HMM approach and dueling-HMM approach.

CHAPTER 5

Threshold-Based HMM Approach

This chapter will discuss the threshold-based HMM approach.

5.1 Threshold-based HMM Implementation

Before using a dueling-HMM approach to do masquerade detection, we first need to compute how effectively a threshold-based HMM approach is for masquerade detection. Although past research has shown that a threshold-based HMM approach is effective in masquerade detection [28] [29] [9], we would still like to verify that it is a viable approach in this project. For this experiment, we only use a threshold-based HMM benign-only approach. The lack of masquerade data to use as training HMMs makes us come to the conclusion that it would not be worthwhile to try doing a threshold-based HMM masquerade-only approach.

5.1.1 Threshold-Based General Implementation Scheme

This section discusses the general implementation approach that uses a combination of different scripts written in various scripting languages. Figure 6 graphically displays the flow of the threshold-based HMM approach. We utilize the Java HMM library (JAHMM) to create the training HMMs by using the k-means learning option. Then we call a smoothing script to smooth the training HMMs so that the training HMMs cover the probability distribution of unseen sequences, so that the training HMMs would not have any "NaN" results. We then use the Estimator to compute the probability between the test data sequence to the training HMMs. Once all the probabilities have been computed, we translate the probabilities using a threshold value to determine whether the test sequence is masqueraded or not.

Figure 6: General Implementation Scheme Flow

5.1.2 Threshold-Based HMM Testing

For this experiment, we are verifying that we can correctly identify each user’s test sequences as masqueraded or not masqueraded. First, we set up the files to use in our experiment. Each user has a training sequence file that contains 5,000 lines of Linux commands that are classified as benign. We then split the remaining 10,000 lines into 100 files, each containing 100 lines of Linux command, to be our test sequence files. Afterwards, we associate each Linux command to an integer value, and convert the sequence files into a sequence of integer values so that it is compatible with the JAHMM library we use to generate our training HMMs.

Once the files are all set up, we follow the implementation scheme in Section 5.1.1 to generate our results in Section 5.2.

When we compute the results to determine whether a test data sequence is masqueraded or not masqueraded, we must consider a threshold ϵ . If probability, p , is less than equal to the threshold, ϵ , we consider that test data sequence to be non-masqueraded. Otherwise, if probability, p , is greater than the threshold, ϵ , we consider that test data sequence to be masqueraded. Adjusting the threshold value, we are able to produce results with an overall accuracy of 95.38%, which captures all the non-masqueraded test data.

5.2 Threshold-base HMM results

Table 5 displays the results from the threshold-based HMM benign-only approach.

Through trial and error in adjusting the threshold, we are able to improve the accuracy of each user. The trade-off is that as the threshold increases, the false positives decrease, while the amount of false negatives increase. For example, User 9 and User 43, we were able to get an accuracy of around the 70 percentile; however, the cost was letting through the majority of the masqueraded test data sequences.

Figure 7 displays the receiver operating characteristic curve (ROC) for the threshold-based HMM approach. We see that with a threshold-based HMM approach, we are able to produce more of a bell curve than compared to Schonlau’s ROC, Figure 8.

Table 6 shows the results of using a threshold with increments of 0.5. For the full table with the threshold incrementing by 0.1 see Table B.15 in Appendix B. Looking at Table 6, we see that with a threshold of around 6.0, the false positive rate drops to less than 10%, while the false negative rate increase to 52%. This trade-off, however, increases the accuracy of our results. Eventually, with a threshold of 14.0 and greater, we get a false positive rate of 0%, but let through 100% of the masqueraded data to get a high accuracy of around 95%.

Table 5: Benign-Only Threshold-Based HMM Results

User	Accuracy	TP	FP	TN	FN	User	Accuracy	TP	FP	TN	FN
User 1	97%	0	3	97	0	User 2	97%	1	1	96	2
User 3	90%	3	2	87	8	User 4	96%	1	3	95	1
User 5	94%	0	6	94	0	User 6	95%	0	5	95	0
User 7	86%	5	6	81	8	User 8	90%	0	10	90	0
User 9	73%	5	8	68	19	User 10	81%	13	19	68	0
User 11	93%	0	7	93	0	User 12	76%	0	18	76	6
User 13	66%	0	34	66	0	User 14	96%	0	4	96	0
User 15	80%	2	16	78	4	User 16	35%	4	59	31	6
User 17	96%	0	4	96	0	User 18	90%	4	8	86	2
User 19	97%	0	3	97	0	User 20	78%	0	22	78	0
User 21	78%	0	22	78	0	User 22	96%	0	4	96	0
User 23	98%	1	2	97	0	User 24	96%	20	3	76	1
User 25	88%	1	4	87	8	User 26	84%	6	9	78	7
User 27	91%	0	9	91	0	User 28	92%	1	6	91	2
User 29	82%	0	17	82	1	User 30	99%	3	1	96	0
User 31	98%	0	2	98	0	User 32	99%	0	1	99	0
User 33	80%	0	20	80	0	User 34	85%	3	6	82	9
User 35	94%	0	5	94	1	User 36	92%	5	7	87	1
User 37	91%	0	7	91	2	User 38	88%	5	8	83	4
User 39	84%	0	16	84	0	User 40	89%	0	11	89	0
User 41	92%	1	6	91	2	User 42	70%	11	21	59	9
User 43	79%	1	6	78	15	User 44	97%	6	3	91	0
User 45	83%	1	13	82	4	User 46	100%	4	0	96	0
User 47	89%	0	11	89	0	User 48	96%	2	4	94	0
User 49	91%	0	9	91	0	User 50	95%	0	5	95	0
Total	88.04%	109	476	4293	122						

Summary: This shows the breakdown for a threshold value of 6 where we start to see lower amounts of false positives, with higher false negatives for some users.

Figure 7: Threshold ROC Plot

Figure 8: Schonlau ROC Plot [27]

Table 6: Benign-Only Threshold-Based HMM Threshold Results

Threshold	False Positives	False Positive (%)	False Negative	False Negative (%)	Accuracy
1.0	4609/4769	96.64%	0/231	0.0%	7.82%
1.5	4384/4769	91.93%	1/231	0.43%	12.3%
2.0	4004/4769	83.96%	8/231	3.46%	19.76%
2.5	3560/4769	74.65%	16/231	6.93%	28.48%
3.0	2979/4769	62.47%	29/231	12.55%	39.84%
3.5	2368/4769	49.65%	44/231	19.05%	51.76%
4.0	1767/4769	37.05%	57/231	24.68%	63.52%
4.5	1287/4769	26.99%	69/231	29.87%	72.88%
5.0	899/4769	18.85%	92/231	39.83%	80.18%
5.5	628/4769	13.17%	106/231	45.89%	85.32%
6.0	476/4769	9.98%	122/231	52.81%	88.04%
6.5	370/4769	7.76%	124/231	53.68%	90.12%
7.0	292/4769	6.12%	130/231	56.28%	91.56%
7.5	255/4769	5.35%	139/231	60.17%	92.12%
8.0	220/4769	4.61%	145/231	62.77%	92.7%
8.5	220/4769	4.61%	145/231	62.77%	92.7%
9.0	179/4769	3.75%	146/231	63.2%	93.5%
9.6	179/4769	3.75%	146/231	63.2%	93.5%
10.0	179/4769	3.75%	146/231	63.2%	93.5%
11.0	179/4769	3.75%	146/231	63.2%	93.5%
12.0	179/4769	3.75%	146/231	63.2%	93.5%
13.0	179/4769	3.75%	146/231	63.2%	93.5%
14.0	0/4769	0.0%	231/231	100.0%	95.38%
15.0	0/4769	0.0%	231/231	100.0%	95.38%

Summary: This is a snippet showing the False Positive Rate, False Negative Rate, and the overall accuracy of the threshold-based HMM masquerade detection results using a threshold.

CHAPTER 6

Dueling-HMM Approach

6.1 Dueling-HMM Implementation

This section discusses the dueling-HMM approach and how we use it on the Schonlau data set for masquerade detection and user identification.

The dueling-HMM approach utilizes multiple HMMs to try to classify which user the test sequence resembles and whether or not it is masqueraded as that user or not. In order for this approach to provide accurate results, the data used for training the multiple HMMs must be accurate and have a sufficient amount of data. Not all users contain masqueraded data in their subsequent test sequences in the Schonlau data set. The following sections discuss the two different experiments, detecting masqueraded data and user classification, and the workaround used to cover for the masqueraded HMMs when there is not any masqueraded data for a particular user.

This approach follows has a similar flow as the implementation scheme discussed in the threshold-based HMM approach in Section 5.1.1. Before using dueling-HMMs for user classification, we need to first verify that the approach is viable for masquerade detection.

6.1.1 Dueling-HMM: Detecting Masqueraded Data Implementation

For this experiment, we are verifying that we can correctly identify each user's test data sequences as masqueraded or not masqueraded. We create the training HMMs using the same approach discussed in the General Implementation Scheme in Section 5.1.1. However, each user would have two training HMMs, one non-masqueraded HMM and one masqueraded HMM, and 100 test data sequences. To compensate for the lack of masqueraded data to use

as the masqueraded training HMM, we decide to use a "fifty-fold" cross validation technique to create a masqueraded training HMM for each user.

Not every user has masqueraded test data. The users that do have masqueraded test data, the amount of masqueraded test data varies for each of those users. For the users that don't have any masqueraded data to use as training, we combine all the masqueraded test data from all the users to create a masqueraded HMM out of those masqueraded data. For the users that do contain masqueraded data, it would not make sense to use that user's masqueraded test data as part of the masqueraded training data. Therefore, we apply the cross-validation technique and create a masqueraded training HMM that combines all the masqueraded test data from all the other users, except for the user that we are creating the masqueraded training HMM for.

To try and improve the results obtained from the dueling-HMM approach, we introduce a bias. To refine the dueling-HMM approach for masquerade detection, we want to adjust the false positive:false negative ratio of our results by introducing a bias to the approach [11]. Figure 9 graphically provides the flow of the dueling-HMM masquerade detection flow.

The bias, ϕ , in this experiment that we adjust to refine our results is the difference, d , between the probabilities computed between the p_1 , probability between test data sequence and benign-only training HMM, and p_2 , probability between test data sequence and masquerade-only training HMM. If the difference, d , is less than equal to the bias, ϕ , we consider that test data sequence to be non-masqueraded. Otherwise, if distance, d , is greater than the threshold, ϕ , we consider that test data sequence to be masqueraded. Adjusting the bias value, we are able to produce results with an overall accuracy 95.34%.

Figure 9: Dueling-HMM Masquerade Detection Flow

6.2 Dueling-HMM: User Classification Implementation

For the user classification experiment, we are trying to verify if we can correctly classify which user the test data belongs to. For this approach, we also create the training HMMs using the approach discussed in the implementation scheme in Section 5.1.1. In this dueling-HMM user classification approach, we decide not to use the masqueraded test data or create training HMMs from the masqueraded data, because whether or not the test data we use is masqueraded as a particular user in this experiment is irrelevant.

This approach follows similarly to the threshold-based HMM benign-only approach, in that we only use the 50 benign-only training data for the 50 training HMMs. The difference though is how we go about computing the probabilities. For the threshold-based HMM benign-only approach, we only compute the probabilities of the 100 test data sequences to

one training HMM, it's corresponding user's training HMM. For the dueling-HMM masquerade detection approach, we computed the probabilities of the 100 test data sequences to two training HMM, it's corresponding user's benign training HMM and the cross-validated masqueraded training HMM. For this approach, we are testing every single non-masqueraded test data sequence to every single training HMM. Figure 10 graphically provides the flow of the dueling-HMM user classification flow.

Figure 10: Dueling-HMM User Classification Flow

So in total, we have 4769 non-masqueraded test sequences. Each of those test sequences would have to compute the probabilities between all 50 training HMMs. We then find the lowest probability the test sequence computed against and classify the test data sequence to that training HMM.

To try and improve the results obtained from the dueling-HMM user classification approach, we introduce a confidence factor. To compute the confidence factor, for each user, we remove the last 1000 lines of Linux commands from the training data and split them into 10 blocks of data sequences consisting of 100 commands each. We then generate the smoothed benign training HMM for each user using the remaining 4000 lines of Linux commands from the training data. Using the Estimator, we compute the probabilities of the 10 blocks that we split from the training data to the benign training HMM. These 10 probabilities are the score samples for that particular user.

Following the approach for dueling-HMM user classification, we compute the probabilities for all the benign test data sequences. To determine which user the test data sequence belongs to we compute the confidence factor for all 50 probabilities for each test sequence. Each user has N different score samples. In our experiment we compute 10 score samples. Now given some unknown user's data, the test data sequences, we obtain the probability, x , for this sequence to a particular HMM. If x is closer to 0 than n of the N scores from the score samples, then we can say that the confidence factor for this data matching this user is n / N .

6.3 Dueling-HMM Testing

The dueling-HMM approach has a similar setup as the threshold-based HMM approach. However, instead of using only one training HMM, we have two training HMMs, one for benign data and one consisting of masqueraded data from all the users using the "fifty-fold" cross validation technique. We use the "fifty-fold" cross validation technique, because not all users contain masquerade data or have enough masquerade data to use as a test data. So with 2 training HMMs and 100 test data sequences, we compute the probabilities between each test data sequence to the two training HMMs. This would produce 50 files, each containing

200 probabilities, 100 to the benign HMM and 100 to the masqueraded HMM. We then compare the two values obtained, the smaller of the two values would determine whether the test data sequence is masqueraded or not. The overall accuracy obtained averaged to be about 81.96%. Though the overall averages came to around a 8% decrease from the threshold-based HMM approach with a threshold value of 6, the dueling-HMM approach had a trade-off of higher false positives while detecting slightly more masqueraded data.

We then introduce the bias to see if we can further improve the accuracy. We determine whether a test data sequence is masqueraded or not by comparing the difference between the two probabilities computed and observe whether it is less than the bias we introduce. With the introduction of the bias, we are able to achieve a maximum overall accuracy of about 95%, which is about the same as when we introduce the threshold in the threshold-based HMM approach.

With the dueling-HMM approach, we also experimented with user classification. We did this experiment only with the benign data, because we are only trying to see if this approach can correctly classify which test data sequence belong to which user. So we would have around 50 training HMMs, one for each user using the user's training data, and the benign-only test data sequence, which is around 4769 test sequences. We then compute the distance of each test data sequence to the 50 training HMMs. Each test data sequence would have 50 distances associated with it. The lowest value would determine which user that test data sequence would classify as. The overall accuracy obtained was 27.18%.

We introduce a confidence factor to the dueling-HMM user classification approach to see if we can improve the results. The results obtained was an overall accuracy of 14.52%. The low overall accuracy is due to the fact that there were many ties for the confidence factor, so we could not clearly distinguish the user the test data sequence belongs to.

6.4 Dueling-HMM Results

This section discusses the results obtained from the dueling-HMM approach for masquerade detection and user classification.

6.4.1 Dueling-HMM Masquerade Detection Results

Table 7 breaks down the statistics for each user using the dueling-HMM approach without introducing a bias.

Compared to the threshold-based HMM approach with a threshold value of 6, we see an overall decrease in the accuracy using the dueling-HMM approach. If we look at the true positive column (TP), we see that the dueling-HMM results is able to classify a higher amount than the threshold-based HMM result. Although we get an slight decrease in the overall accuracy using the dueling-HMM approach, the tradeoff is that the dueling-HMM approach is able to be obtain more true positives and less amounts of false negatives; however, with higher amounts of false positives than the threshold-based HMM benign-only approach.

6.4.2 Dueling-HMM Bias Results

Figure 11 graphically shows the ROC curve for the dueling-HMM approach using the bias.

Table 8 shows the results of using a bias with increments of 0.5. For the full table with the bias incrementing by 0.1 see Table C.17 in Appendix C. Looking at Table 8, we see that with a bias of around 4.0 - 5.0, the false positive rate drops to less than 10.0%, while the false negative rate increase to about 60.0%. This trade-off, however, increases the accuracy of our results. Eventually, with a bias of 11.0 and greater, we get a false positive rate of 0.0%, but we let through around 100% of the masqueraded data to get a high accuracy of

Table 7: Dueling-HMM Results

User	Accuracy	TP	FP	TN	FN	User	Accuracy	TP	FP	TN	FN
User 1	90%	0	10	90	0	User 2	96%	3	4	93	0
User 3	90%	7	6	83	4	User 4	78%	2	22	76	0
User 5	89%	0	11	89	0	User 6	86%	0	14	86	0
User 7	79%	8	16	71	5	User 8	84%	0	16	84	0
User 9	69%	9	16	60	15	User 10	76%	13	24	63	0
User 11	82%	0	18	82	0	User 12	72%	0	22	72	6
User 13	59%	0	41	59	0	User 14	90%	0	10	90	0
User 15	71%	1	24	70	5	User 16	33%	5	62	28	5
User 17	87%	0	13	87	0	User 18	88%	5	11	83	1
User 19	85%	0	15	85	0	User 20	78%	0	22	78	0
User 21	83%	0	17	83	0	User 22	84%	0	16	84	0
User 23	95%	1	5	94	0	User 24	93%	20	6	73	1
User 25	82%	8	17	74	1	User 26	82%	6	11	76	7
User 27	89%	0	11	89	0	User 28	82%	3	18	79	0
User 29	73%	0	26	73	1	User 30	99%	3	1	96	0
User 31	93%	0	7	93	0	User 32	99%	0	1	99	0
User 33	69%	0	31	69	0	User 34	76%	3	15	73	9
User 35	92%	0	7	92	1	User 36	89%	6	11	83	0
User 37	80%	0	18	80	2	User 38	84%	5	12	79	4
User 39	76%	0	24	76	0	User 40	85%	0	15	85	0
User 41	80%	2	19	78	1	User 42	69%	11	22	58	9
User 43	73%	2	13	71	14	User 44	90%	6	10	84	0
User 45	77%	1	19	76	4	User 46	96%	4	4	92	0
User 47	74%	0	26	74	0	User 48	84%	2	16	82	0
User 49	85%	0	15	85	0	User 50	83%	0	17	83	0
Total	81.96%	136	807	3962	95						

Summary: Using the dueling-HMM approach, we see a slight decrease in the overall accuracy, 81.96%. However, we see lower amounts of false negatives than in the threshold-based HMM approach.

around 95%.

6.4.3 Dueling-HMM User Classification Results

Table 9 displays the results of the dueling-HMM user classification experiment. The table consists of the amount of correctly classified test data sequence (Match), the total number of test data sequences associated with each user (Total), and the accuracy of the

Table 8: Dueling-HMM Bias Result

Bias	False Positives	False Positive (%)	False Negative	False Negative (%)	Accuracy
1.0	3670/4769	76.96%	60/231	25.97%	25.4%
1.5	3088/4769	64.75%	78/231	33.77%	36.68%
2.0	2445/4769	51.27%	100/231	43.29%	49.1%
2.5	1898/4769	39.8%	122/231	52.81%	59.6%
3.0	1313/4769	27.53%	133/231	57.58%	71.08%
3.5	1013/4769	21.24%	136/231	58.87%	77.02%
4.0	630/4769	13.21%	144/231	62.34%	84.52%
4.5	440/4769	9.23%	147/231	63.64%	88.26%
5.0	303/4769	6.35%	148/231	64.07%	90.98%
5.5	202/4769	4.24%	148/231	64.07%	93.0%
6.0	184/4769	3.86%	149/231	64.5%	93.34%
6.5	179/4769	3.75%	149/231	64.5%	93.44%
7.0	173/4769	3.63%	149/231	64.5%	93.56%
7.5	140/4769	2.94%	151/231	65.37%	94.18%
8.0	101/4769	2.12%	169/231	73.16%	94.6%
8.5	63/4769	1.32%	178/231	77.06%	95.18%
9.0	33/4769	0.69%	192/231	83.12%	95.5%
9.5	11/4769	0.23%	222/231	96.1%	95.34%
10.0	6/4769	0.13%	227/231	98.27%	95.34%
11.0	0/4769	0.0%	230/231	99.57%	95.4%
12.0	0/4769	0.0%	231/231	100.0%	95.38%

Summary: This is a snippet showing the False Positive Rate, False Negative Rate, and the overall accuracy of the dueling-HMM masquerade detection results using a bias.

test data sequences correctly classified (Accuracy).

It is interesting to note that the dueling-HMM approach does not produce very desirable results for user classification as shown in Table 9. It seems to have trouble distinguishing which test data sequence belongs to which user, only achieving an overall accuracy of 27.18%.

6.4.4 Dueling-HMM User Classification Confidence Factor Results

Table 10 displays the results of the dueling-HMM user classification approach using confidence factors. The extra column, Ties, shows the number of test sequences in which one of the ties for the highest confidence factor was its corresponding user. From Table 10, we see that using confidence factors results in many ties, indicating that the approach has

Figure 11: Dueling-HMM Bias ROC Plot

difficulty in clearly distinguishing the test data sequences to it's corresponding user.

Table 9: User Classification Results

User	Match	Total	Accuracy	User	Match	Total	Accuracy
User 1	28	100	28.0%	User 2	56	97	57.73%
User 3	42	89	47.19%	User 4	25	98	25.51%
User 5	17	100	17.0%	User 6	45	100	45.0%
User 7	10	87	11.49%	User 8	67	100	67.0%
User 9	27	76	35.53%	User 10	22	87	25.29%
User 11	4	100	4.0%	User 12	23	94	24.47%
User 13	31	100	31.0%	User 14	12	100	12.0%
User 15	4	94	4.26%	User 16	10	90	11.11%
User 17	15	100	15.0%	User 18	66	94	70.21%
User 19	0	100	0.0%	User 20	0	100	0.0%
User 21	42	100	42.0%	User 22	19	100	19.0%
User 23	3	99	3.03%	User 24	17	79	21.52%
User 25	21	91	23.08%	User 26	31	87	35.63%
User 27	14	100	14.0%	User 28	18	97	18.56%
User 29	5	99	5.05%	User 30	96	97	98.97%
User 31	31	100	31.0%	User 32	53	100	53.0%
User 33	12	100	12.0%	User 34	5	88	5.68%
User 35	62	99	62.63%	User 36	59	94	62.77%
User 37	36	98	36.73%	User 38	12	91	13.19%
User 39	12	100	12.0%	User 40	26	100	26.0%
User 41	0	97	0.0%	User 42	12	80	15.0%
User 43	12	84	14.29%	User 44	17	94	18.09%
User 45	47	95	49.47%	User 46	50	96	52.08%
User 47	49	100	49.0%	User 48	25	98	25.51%
User 49	8	100	8.0%	User 50	0	100	0.0%
Total	1298	4769	27.18%				

Summary: The dueling-HMM provides quite low accuracy in trying to classify which user a test data sequence belongs.

Table 10: User Classification Confidence Factor Results

User	Match	Ties	Total	Accuracy	User	Match	Ties	Total	Accuracy
User 1	0	28	100	0.0%	User 2	22	0	97	22.68%
User 3	20	33	89	22.47%	User 4	0	21	98	0.0%
User 5	0	17	100	0.0%	User 6	0	17	100	0.0%
User 7	0	10	87	0.0%	User 8	67	0	100	67.0%
User 9	21	0	76	27.63%	User 10	24	12	87	27.59%
User 11	16	14	100	16.0%	User 12	1	41	94	1.06%
User 13	9	22	100	9.0%	User 14	0	12	100	0.0%
User 15	2	1	94	2.13%	User 16	6	0	90	6.67%
User 17	16	0	100	16.0%	User 18	66	0	94	70.21%
User 19	6	16	100	6.0%	User 20	5	7	100	5.0%
User 21	0	25	100	0.0%	User 22	18	10	100	18.0%
User 23	5	47	99	5.05%	User 24	11	11	79	13.92%
User 25	15	0	91	16.48%	User 26	5	24	87	5.75%
User 27	6	3	100	6.0%	User 28	9	24	97	9.28%
User 29	0	13	99	0.0%	User 30	67	0	97	69.07%
User 31	22	9	100	22.0%	User 32	0	17	100	0.0%
User 33	17	4	100	17.0%	User 34	9	29	88	10.23%
User 35	6	54	99	6.06%	User 36	14	18	94	14.89%
User 37	32	0	98	32.65%	User 38	6	14	91	6.59%
User 39	15	9	100	15.0%	User 40	8	10	100	8.0%
User 41	9	11	97	9.28%	User 42	5	2	80	6.25%
User 43	0	40	84	0.0%	User 44	0	10	94	0.0%
User 45	30	1	95	31.58%	User 46	0	19	96	0.0%
User 47	27	0	100	27.0%	User 48	28	4	98	28.57%
User 49	34	27	100	34.0%	User 50	14	2	100	14.0%
Total	693	688	4769	14.52%					

Summary: Using the Confidence Factor, we obtain an even lower score due to the fact that there are many ties; hence, we are unable to clearly distinguish which user the test sequence belongs to.

CHAPTER 7

Conclusion

Masqueraders are users that try to impersonate their victims. In this research, we explore the heavy hitter approach, threshold-based HMM approach, and dueling-HMM approach to see how effectively these approaches can detect masqueraders using the Schonlau data set. We conclude that the heavy hitter approach is not a very viable approach to do masquerade detection. On the other hand, the threshold-based HMM approach and dueling-HMM approach produce desirable results through introducing a threshold and bias.

The heavy hitter approach that utilizes the top k elements of a sequence of data did not provide an adequate representation of a user. By trying to see how many of the test data sequences' top k elements matches that of the top k elements of the training data sequence provided insignificant data to determine whether the test sequence is masqueraded or not. Applying the Levenshtein distance to compute the similarity between the top k elements of the training sequence and test sequence showed a slight increase in the results; however, the results were still far from satisfactory with 90% of the users getting below an overall accuracy of 50%. While able to capture most of the masqueraded data, users would be flooded with false alarms.

The threshold-based HMM proves to be an adequate approach to doing masquerade detection as we have verified in our project to the results seen in past researches. We obtain an overall accuracy of 95.38% with a threshold value of 14.0 or higher. To see if the accuracy could be further improved, we experiment with the the dueling-HMM approach.

Using the dueling-HMM approach, we also obtain an overall accuracy of 95.38% with a bias value of 9.5 and higher. We see that in both threshold-based HMM approach and

dueling-HMM approach, we are able to capture all the non-masqueraded test data sequences, while 100% of the masqueraded test data sequences are not captured, yet still resulting in an overall accuracy of 95.38%.

We also experimented with classifying users with the dueling-HMM approach. Contrary to the results we see for using the dueling-HMM approach for masquerade detection, we see an overall accuracy of 27.18% for the dueling-HMM approach for user classification. Furthermore, when we introduce the confidence factors into the dueling-HMM approach for user classification, we see an overall accuracy of 14.52% due to the fact that the dueling-HMM approach has difficulty it clearly distinguishing which test data sequence belongs to which user because of the many ties when computing the confidence factor. This may also be due to the fact that we reduced the amount of training data used to generate the HMM so that we can obtain score samples to compute the confidence factor.

In this project, we create HMMs from the Schonlau training data for the threshold-based HMM approach and dueling-HMM approach. In M. Stamp and L. Huang's article [5], we see that using a profile hidden Markov model (PHMM) has the advantage over HMMs using a limited training data set, where the PHMM's feature is utilizing the order in which a user issues commands. To see if we can further improve the dueling-HMM approach, it would be interesting to see if integrating PHMMs would improve the accuracy, using a limited training data set and with the Schonlau data set.

LIST OF REFERENCES

- [1] A. Metwally and D. Agrawal and A. Abbadi, Efficient Computation of Frequent and Top- k Elements in Data Streams, Department of Computer Science, University of California, Santa Barbara, 2005
- [2] M. Bertacchini and P. I. Fierens, A Survey on Masquerader Detection Approaches. CIBSI, 2009
[http://www.criptored.upm.es/cibsi/cibsi2009/docs/Papers/CIBSI-Dia2-Sesion5\(2\).pdf](http://www.criptored.upm.es/cibsi/cibsi2009/docs/Papers/CIBSI-Dia2-Sesion5(2).pdf)
- [3] M. Schonlau and W. DuMouchel and W. H. Ju and A. F. Karr and M. Theus and Y. Vardi. Computer Intrusion: Detecting Masquerades. Stat Sci. 16(1): 58-74, Feb. 2001
- [4] M. Schonlau, Masquerading User Data. In Matthias Schonlaus (Matt Schonlau) Homepage. Accessed Oct. 2015
<http://schonlau.net/>
- [5] M. Stamp and L. Huang, Computers & Security, *Masquerade detection using profile hidden markov models*, Vol. 30, issue 8, pp. 734-747, 2011
- [6] T. Locher, Finding Heavy Distinct Hitters in Data Streams, ETH Zurich, IBM Research, 2011
disco.ethz.ch/publications/spaa1-locher_239.pdf
- [7] R. F. Erbacher and S. Prakash and C. L. Clarr and J. Couraud, Intrusion Detection: Detecting Masquerade Attacks Using UNIX Command Lines, Utah State University, 2009
<http://digital.cs.usu.edu/~erbacher/publications/MasqueradeDetectionConference.pdf>
- [8] G. Cormod, Finding Frequent Items in Data Streams, 2008
dmac.rutgers.edu/Workshops/WGUNifyingTheory/Slides/cormode.pdf
- [9] Z. Ghahramani, International Journal of Pattern Recognition and Artificial Intelligence, *An Introduction to Hidden Markov Models and Bayesian Networks*, Vol. 15, issue 1, pp. 9-42, 2001
<http://mlg.eng.cam.ac.uk/zoubin/papers/ijprai.pdf>
- [10] J. Francois, JAHMM - An implementation of Hidden Markov Models in Java. Accessed Feb 4, 2010
<https://code.google.com/p/jahmm/>

- [11] A. Kalbhor, A Tiered Approach to Detect Metamorphic Malware With Hidden Markov Models, Department of Computer Science, San Jose State University, 2014
- [12] B. K. Szymanski and Y. Q. Zhang, Man and Cybernetics Information Assurance Workshop, *Recursive Data Mining for Masquerade Detection and Author Identification*, Proc. 5th IEEE System, pp. 424-431, 2004
- [13] C. G. Nevill-Manning, I. H. Witten, Identifying hierarchical structure in sequences: a linear-time algorithm. *Journal of Artificial Intelligence Research* 7, pp. 67-82, 1997
<https://www.jair.org/media/374/live-374-1630-jair.pdf>
- [14] M. Schonlau, W. Dumouchel, W. H. Ju, A. F. Karr, M. Theus, Y. Vardi, Computer intrusion: detecting masquerades. *Statistical Science* 16, pp. 58-74, 2001
<http://www.schonlau.net/publication/01statscigalley.pdf>
- [15] W. H. Ju and Y. Vardi, A hybrid high-order Markov chain model for computer intrusion detection. Technical report 92, National Institute Statist. Sci. 1999
<http://www.niss.org/sites/default/files/technicalreports/tr92.pdf>
- [16] A. McCallum and K. Nigam, A comparison of event models for naive bayes text classification, 1998
- [17] T. Lane, C. E. Brodley, An application of machine learning to anomaly detection. In *Proceedings of the 20th National Information Systems Security Conference*, pp. 366-380, 1997
- [18] S. B. Böckler and A. Bateman, An introduction to hidden markov models. *Current protocols in bioinformatics* / editorial board, Andreas D. Baxeavanis ... [et al.] Appendix 3, 2007
- [19] A. Kothari, Defeating Masquerade Detection, Department of Computer Science, San Jose State University, San Jose, 2012
- [20] S. Coull, J. Branch, B. Szymanski, E. Breimer, Intrusion Detection: A Bioinformatics Approach, 2003
<https://www.acsac.org/2003/papers/35.pdf>
- [21] B. M. Reshmi and S. S. Manvi, Masquerade Detection by Using Activity Patterns, EC2ND, pp. 241-250, 2005
- [22] H. S. Kim, S. D. Cha, Empirical evaluation of SVM-based masquerade detection using UNIX commands, *Computer & Security* 24, pp. 160-168, 2005
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.97.8452&rep=rep1&type=pdf>

- [23] T. H. Austin, E. Filiol, S. Josse, and M. Stamp, Exploring hidden Markov models for virus analysis: A semantic approach. In System Sciences (HICSS), 2013 46th Hawaii International Conference on IEEE, pp. 5039-5048, Jan. 2005
- [24] G. Cormode, F. Korn, S. Muthukrishnan, D. Srivastava, Finding Hierarchical Heavy Hitters in Data Streams, ACM Transactions on Database Systems, Vol. V, No. N, October 2007 <http://dimacs.rutgers.edu/~graham/pubs/papers/ckms-hhh.pdf>
- [25] S. Boodidhi, Using Smoothing Techniques to Improve the Performance of Hidden Markov's Model, Master's report, School of Computer Science, Howard R. Hughes College of Engineering, 2011
<http://digitalscholarship.unlv.edu/cgi/viewcontent.cgi?article=2008&context=thesesdissertations>
- [26] G. .D. Xu, Y. Zong, and Z. L. Yang, *Applied Data Mining*, CRC Press, First Edition, Jun. 17, 2013
- [27] M. Schonlau, W. DuMouchel, W. H. Ju, A. F. Karr, M. Theus, and Y. Vardi, Computer Intrusion: Detecting Masquerades, *Statistical Science*, Vol. 16, No. 1, 1-17, 2001
<http://www.schonlau.net/publication/01statscigalley.pdf>
- [28] R. Saradha, Malware Analysis using Profile Hidden Markov Models and Intrusion Detection in a Stream Learning Setting, Supercomputer Education and Research Centre, Indian Institute of Science, May 2014
http://www.serc.iisc.ernet.in/graduation-theses/Thesis_saradha.pdf
- [29] S. K. Sharma, International Journal of Computer Applications, *Intrusion Detection using Hidden Markov Model*, Volume 115, No. 4, April 2015
<http://research.ijcaonline.org/volume115/number4/pxc3902264.pdf>
- [30] B. Bigi, Using Kullback-Leibler Distance for Text Categorization, CLIPS-IMAG Laboratory, F. Sebastini (Ed.): ECIR 2003, LNCS 2633, pp. 305-319, 2003
<http://users.softlab.ntua.gr/facilities/public/AD/Text/%20Categorization/Using%20Kullback-Leibler%20Distance%20for%20Text%20Categorization.pdf>
- [31] D. C. D'Elia, Mining Heavy Hitters with Space-Saving, Dept. of Computer, Control and Management Engineering, Sapienza University of Rome, August 11, 2013
<http://www.dis.uniroma1.it/~delia/files/docs/spacesaving-report.pdf>
- [32] A. Wagner, T. Dübendorfer, L. Hämmerle, B. Plattner, Identifying P2P Heavy-Hitters from Network-Flow Data, Software Engineering Institute, Carnegie Mellon University, September 2005
<http://www.cert.org/flocon/2005/presentations/Wagner-HeavyHitters-FloCon2005.pdf>

- [33] T. H. Austin, "JAHMM Utils", Accessed 2015.
<https://github.com/taustin/JahmmUtils>

APPENDIX A

Heavy Hitter Space Saving Results

Table A.11: HH - Top 10 - 20%

User	Percentage	TP	FP	TN	FN	User	Percentage	TP	FP	TN	FN
User 1	79%	0	21	79	0	User 2	98%	3	2	95	0
User 3	91%	8	6	83	3	User 4	46%	2	54	44	0
User 5	95%	0	5	95	0	User 6	93%	0	7	93	0
User 7	71%	12	28	59	1	User 8	24%	0	76	24	0
User 9	82%	17	12	65	6	User 10	51%	13	49	38	0
User 11	69%	0	31	69	0	User 12	40%	0	55	40	5
User 13	62%	0	38	62	0	User 14	28%	0	72	28	0
User 15	30%	5	69	25	1	User 16	38%	8	60	30	2
User 17	82%	0	18	82	0	User 18	47%	5	52	42	1
User 19	95%	0	5	95	0	User 20	53%	0	47	53	0
User 21	90%	0	10	90	0	User 22	42%	0	58	42	0
User 23	62%	1	38	61	0	User 24	74%	20	25	54	1
User 25	39%	9	61	30	0	User 26	66%	5	27	61	7
User 27	24%	0	76	24	0	User 28	62%	3	38	59	0
User 29	35%	1	65	34	0	User 30	38%	3	62	35	0
User 31	60%	0	40	60	0	User 32	72%	0	28	72	0
User 33	97%	0	3	97	0	User 34	49%	9	49	40	2
User 35	89%	1	11	88	0	User 36	82%	3	15	79	3
User 37	79%	1	20	78	1	User 38	90%	5	8	85	2
User 39	76%	0	24	76	0	User 40	82%	0	18	82	0
User 41	24%	3	76	21	0	User 42	61%	7	27	54	12
User 43	80%	6	11	74	9	User 44	83%	5	17	78	0
User 45	84%	2	14	82	2	User 46	96%	4	4	92	0
User 47	80%	0	20	80	0	User 48	92%	2	8	90	0
User 49	55%	0	45	55	0	User 50	76%	0	24	76	0
Total	66.26%	163	1629	3150	58						

Summary: This table shows the results when we monitor the Top 10 Elements and set a threshold value of 2 to determine whether a test is masqueraded or not. The overall accuracy is 66.26%.

Table A.12: HH - Top 10 - 30%

User	Percentage	TP	FP	TN	FN	User	Percentage	TP	FP	TN	FN
User 1	60%	0	40	60	0	User 2	96%	3	4	93	0
User 3	83%	10	16	73	1	User 4	29%	2	71	27	0
User 5	75%	0	25	75	0	User 6	90%	0	10	90	0
User 7	34%	12	65	22	1	User 8	11%	0	89	11	0
User 9	64%	24	36	40	0	User 10	42%	13	58	29	0
User 11	48%	0	52	48	0	User 12	19%	1	76	18	5
User 13	27%	0	73	27	0	User 14	9%	0	91	9	0
User 15	24%	5	75	19	1	User 16	25%	10	75	15	0
User 17	79%	0	21	79	0	User 18	24%	5	75	19	1
User 19	86%	0	14	86	0	User 20	17%	0	83	17	0
User 21	83%	0	17	83	0	User 22	26%	0	74	26	0
User 23	11%	1	89	10	0	User 24	48%	21	52	27	0
User 25	18%	9	82	9	0	User 26	39%	11	59	28	2
User 27	7%	0	93	7	0	User 28	37%	3	63	34	0
User 29	9%	1	91	8	0	User 30	34%	3	66	31	0
User 31	36%	0	64	36	0	User 32	63%	0	37	63	0
User 33	93%	0	7	93	0	User 34	30%	11	69	19	1
User 35	29%	1	71	28	0	User 36	56%	4	42	52	2
User 37	51%	1	48	50	1	User 38	74%	5	24	69	2
User 39	57%	0	43	57	0	User 40	54%	0	46	54	0
User 41	17%	3	82	14	0	User 42	38%	14	55	24	6
User 43	61%	14	37	47	2	User 44	61%	5	39	56	0
User 45	60%	4	40	56	0	User 46	87%	4	13	83	0
User 47	53%	0	47	53	0	User 48	66%	2	34	64	0
User 49	31%	0	69	31	0	User 50	43%	0	57	43	0
Total	46.28%	202	2661	2112	25						

Summary: This table shows the results when we monitor the Top 10 Elements and set a threshold value of 3 to determine whether a test is masqueraded or not. The overall accuracy is 46.28%.

Table A.13: HH - Top 10 - 40%

User	Percentage	TP	FP	TN	FN	User	Percentage	TP	FP	TN	FN
User 1	40%	0	60	40	0	User 2	86%	3	14	83	0
User 3	68%	11	32	57	0	User 4	10%	2	90	8	0
User 5	53%	0	47	53	0	User 6	81%	0	19	81	0
User 7	20%	13	80	7	0	User 8	2%	0	98	2	0
User 9	44%	24	56	20	0	User 10	37%	13	63	24	0
User 11	30%	0	70	30	0	User 12	14%	4	85	10	1
User 13	10%	0	90	10	0	User 14	0%	0	100	0	0
User 15	18%	5	81	13	1	User 16	14%	10	86	4	0
User 17	77%	0	23	77	0	User 18	10%	6	90	4	0
User 19	80%	0	20	80	0	User 20	3%	0	97	3	0
User 21	74%	0	26	74	0	User 22	14%	0	86	14	0
User 23	1%	1	99	0	0	User 24	31%	21	69	10	0
User 25	12%	9	88	3	0	User 26	19%	12	80	7	1
User 27	1%	0	99	1	0	User 28	20%	3	80	17	0
User 29	3%	1	97	2	0	User 30	3%	3	97	0	0
User 31	11%	0	89	11	0	User 32	50%	0	50	50	0
User 33	72%	0	28	72	0	User 34	23%	12	77	11	0
User 35	21%	1	79	20	0	User 36	40%	6	60	34	0
User 37	31%	1	68	30	1	User 38	63%	7	36	56	1
User 39	34%	0	66	34	0	User 40	35%	0	65	35	0
User 41	9%	3	91	6	0	User 42	25%	17	72	8	3
User 43	36%	16	64	20	0	User 44	37%	6	63	31	0
User 45	26%	4	74	22	0	User 46	73%	4	27	69	0
User 47	33%	0	67	33	0	User 48	40%	2	60	38	0
User 49	8%	0	92	8	0	User 50	18%	0	82	18	0
Total	31.2%	220	3432	1340	8						

Summary: This table shows the results when we monitor the Top 10 Elements and set a threshold value of 4 to determine whether a test is masqueraded or not. The overall accuracy is 31.2%.

Table A.14: HH - Top 10 - 50%

User	Percentage	TP	FP	TN	FN	User	Percentage	TP	FP	TN	FN
User 1	21%	0	79	21	0	User 2	71%	3	29	68	0
User 3	49%	11	51	38	0	User 4	5%	2	95	3	0
User 5	21%	0	79	21	0	User 6	63%	0	37	63	0
User 7	15%	13	85	2	0	User 8	1%	0	99	1	0
User 9	33%	24	67	9	0	User 10	27%	13	73	14	0
User 11	11%	0	89	11	0	User 12	8%	5	92	3	0
User 13	2%	0	98	2	0	User 14	0%	0	100	0	0
User 15	11%	5	88	6	1	User 16	12%	10	88	2	0
User 17	65%	0	35	65	0	User 18	7%	6	93	1	0
User 19	67%	0	33	67	0	User 20	0%	0	100	0	0
User 21	64%	0	36	64	0	User 22	9%	0	91	9	0
User 23	1%	1	99	0	0	User 24	22%	21	78	1	0
User 25	11%	9	89	2	0	User 26	15%	13	85	2	0
User 27	1%	0	99	1	0	User 28	6%	3	94	3	0
User 29	1%	1	99	0	0	User 30	3%	3	97	0	0
User 31	6%	0	94	6	0	User 32	39%	0	61	39	0
User 33	56%	0	44	56	0	User 34	16%	12	84	4	0
User 35	8%	1	92	7	0	User 36	28%	6	72	22	0
User 37	15%	2	85	13	0	User 38	44%	9	55	35	0
User 39	16%	0	84	16	0	User 40	9%	0	91	9	0
User 41	4%	3	96	1	0	User 42	23%	20	77	3	0
User 43	20%	16	80	4	0	User 44	9%	6	91	3	0
User 45	14%	5	86	9	0	User 46	53%	4	47	49	0
User 47	21%	0	79	21	0	User 48	18%	2	82	16	0
User 49	2%	0	98	2	0	User 50	4%	0	96	4	0
Total	20.54%	229	3972	798	1						

Summary: This table shows the results when we monitor the Top 10 Elements and set a threshold value of 5 to determine whether a test is masqueraded or not. The overall accuracy is 20.54%.

APPENDIX B

Threshold-Based HMM Threshold Results

Table B.15: Threshold-Based HMM Threshold Result

Threshold	False Positives	False Positive (%)	False Negative	False Negative (%)	Accuracy
1.0	4609/4769	96.64%	0/231	0.0%	7.82%
1.1	4542/4769	95.24%	0/231	0.0%	9.16%
1.2	4513/4769	94.63%	0/231	0.0%	9.74%
1.3	4453/4769	93.37%	0/231	0.0%	10.94%
1.4	4438/4769	93.06%	0/231	0.0%	11.24%
1.5	4384/4769	91.93%	1/231	0.43%	12.3%
1.6	4336/4769	90.92%	1/231	0.43%	13.26%
1.7	4283/4769	89.81%	1/231	0.43%	14.32%
1.8	4207/4769	88.22%	4/231	1.73%	15.78%
1.9	4132/4769	86.64%	4/231	1.73%	17.28%
2.0	4004/4769	83.96%	8/231	3.46%	19.76%
2.1	3952/4769	82.87%	11/231	4.76%	20.74%
2.2	3859/4769	80.92%	12/231	5.19%	22.58%
2.3	3767/4769	78.99%	13/231	5.63%	24.4%
2.4	3702/4769	77.63%	13/231	5.63%	25.7%
2.5	3560/4769	74.65%	16/231	6.93%	28.48%
2.6	3459/4769	72.53%	23/231	9.96%	30.36%
2.7	3346/4769	70.16%	24/231	10.39%	32.6%
2.8	3246/4769	68.06%	27/231	11.69%	34.54%
2.9	3128/4769	65.59%	29/231	12.55%	36.86%
3.0	2979/4769	62.47%	29/231	12.55%	39.84%
3.1	2885/4769	60.49%	31/231	13.42%	41.68%
3.2	2697/4769	56.55%	35/231	15.15%	45.36%
3.3	2556/4769	53.6%	36/231	15.58%	48.16%
3.4	2456/4769	51.5%	42/231	18.18%	50.04%
3.5	2368/4769	49.65%	44/231	19.05%	51.76%
3.6	2243/4769	47.03%	47/231	20.35%	54.2%
3.7	2123/4769	44.52%	49/231	21.21%	56.56%
3.8	1985/4769	41.62%	49/231	21.21%	59.32%
3.9	1886/4769	39.55%	54/231	23.38%	61.2%
4.0	1767/4769	37.05%	57/231	24.68%	63.52%
4.1	1648/4769	34.56%	60/231	25.97%	65.84%
4.2	1539/4769	32.27%	62/231	26.84%	67.98%
4.3	1453/4769	30.47%	65/231	28.14%	69.64%
4.4	1380/4769	28.94%	65/231	28.14%	71.1%
4.5	1287/4769	26.99%	69/231	29.87%	72.88%
4.6	1226/4769	25.71%	72/231	31.17%	74.04%
4.7	1150/4769	24.11%	79/231	34.2%	75.42%
4.8	1044/4769	21.89%	83/231	35.93%	77.46%
4.9	994/4769	20.84%	89/231	38.53%	78.34%
5.0	899/4769	18.85%	92/231	39.83%	80.18%
5.1	846/4769	17.74%	95/231	41.13%	81.18%
5.2	781/4769	16.38%	99/231	42.86%	82.4%
5.3	709/4769	14.87%	105/231	45.45%	83.72%
5.4	674/4769	14.13%	105/231	45.45%	84.42%
5.5	628/4769	13.17%	106/231	45.89%	85.32%
5.6	590/4769	12.37%	111/231	48.05%	85.98%
5.7	551/4769	11.55%	121/231	52.38%	86.56%
5.8	539/4769	11.3%	121/231	52.38%	86.8%
5.9	493/4769	10.34%	122/231	52.81%	87.7%
6.0	476/4769	9.98%	122/231	52.81%	88.04%

Summary: Shows the data of the threshold in increments of 0.1. Shows how the False Positive Percent decreasing as the False Negative Percent increases to get an overall higher accuracy.

Table B.16: Threshold-Based HMM Threshold ResulContinuation

Threshold	False Positives	False Positive (%)	False Negative	False Negative (%)	Accuracy
6.1	457/4769	9.58%	122/231	52.81%	88.42%
6.2	449/4769	9.41%	123/231	53.25%	88.56%
6.3	419/4769	8.79%	123/231	53.25%	89.16%
6.4	395/4769	8.28%	123/231	53.25%	89.64%
6.5	370/4769	7.76%	124/231	53.68%	90.12%
6.6	356/4769	7.46%	127/231	54.98%	90.34%
6.7	356/4769	7.46%	127/231	54.98%	90.34%
6.8	329/4769	6.9%	128/231	55.41%	90.86%
6.9	329/4769	6.9%	128/231	55.41%	90.86%
7.0	292/4769	6.12%	130/231	56.28%	91.56%
7.1	292/4769	6.12%	130/231	56.28%	91.56%
7.2	268/4769	5.62%	137/231	59.31%	91.9%
7.3	268/4769	5.62%	137/231	59.31%	91.9%
7.4	268/4769	5.62%	137/231	59.31%	91.9%
7.5	255/4769	5.35%	139/231	60.17%	92.12%
7.6	255/4769	5.35%	139/231	60.17%	92.12%
7.7	255/4769	5.35%	139/231	60.17%	92.12%
7.8	255/4769	5.35%	139/231	60.17%	92.12%
7.9	220/4769	4.61%	145/231	62.77%	92.7%
8.0	220/4769	4.61%	145/231	62.77%	92.7%
8.1	220/4769	4.61%	145/231	62.77%	92.7%
8.2	220/4769	4.61%	145/231	62.77%	92.7%
8.3	220/4769	4.61%	145/231	62.77%	92.7%
8.4	220/4769	4.61%	145/231	62.77%	92.7%
8.5	220/4769	4.61%	145/231	62.77%	92.7%
8.6	179/4769	3.75%	146/231	63.2%	93.5%
8.7	179/4769	3.75%	146/231	63.2%	93.5%
8.8	179/4769	3.75%	146/231	63.2%	93.5%
8.9	179/4769	3.75%	146/231	63.2%	93.5%
9.0	179/4769	3.75%	146/231	63.2%	93.5%
9.1	179/4769	3.75%	146/231	63.2%	93.5%
9.2	179/4769	3.75%	146/231	63.2%	93.5%
9.3	179/4769	3.75%	146/231	63.2%	93.5%
9.4	179/4769	3.75%	146/231	63.2%	93.5%
9.5	179/4769	3.75%	146/231	63.2%	93.5%
9.6	179/4769	3.75%	146/231	63.2%	93.5%
9.7	179/4769	3.75%	146/231	63.2%	93.5%
9.8	179/4769	3.75%	146/231	63.2%	93.5%
9.9	179/4769	3.75%	146/231	63.2%	93.5%
10.0	179/4769	3.75%	146/231	63.2%	93.5%
11.0	179/4769	3.75%	146/231	63.2%	93.5%
12.0	179/4769	3.75%	146/231	63.2%	93.5%
13.0	179/4769	3.75%	146/231	63.2%	93.5%
14.0	0/4769	0.0%	231/231	100.0%	95.38%
15.0	0/4769	0.0%	231/231	100.0%	95.38%
16.0	0/4769	0.0%	231/231	100.0%	95.38%
17.0	0/4769	0.0%	231/231	100.0%	95.38%
18.0	0/4769	0.0%	231/231	100.0%	95.38%
19.0	0/4769	0.0%	231/231	100.0%	95.38%
20.0	0/4769	0.0%	231/231	100.0%	95.38%

Summary: Shows the data of the threshold in increments of 0.1. Shows how the False Positive Percent decreasing as the False Negative Percent increases to get an overall higher accuracy. Eventually tapering off at around a threshold 14.0 where there are no false positives, but misses nearly all the masqueraded data, yet still resulting in an accuracy of 95.38%.

APPENDIX C

Dueling-HMM Bias Results

Table C.17: Dueling-HMM Bias Result

Bias	False Positives	False Positive (%)	False Negative	False Negative (%)	Accuracy
1.0	3670/4769	76.96%	60/231	25.97%	25.4%
1.1	3549/4769	74.42%	61/231	26.41%	27.8%
1.2	3437/4769	72.07%	66/231	28.57%	29.94%
1.3	3330/4769	69.83%	68/231	29.44%	32.04%
1.4	3200/4769	67.1%	74/231	32.03%	34.52%
1.5	3088/4769	64.75%	78/231	33.77%	36.68%
1.6	2988/4769	62.65%	83/231	35.93%	38.58%
1.7	2895/4769	60.7%	86/231	37.23%	40.38%
1.8	2758/4769	57.83%	91/231	39.39%	43.02%
1.9	2599/4769	54.5%	95/231	41.13%	46.12%
2.0	2445/4769	51.27%	100/231	43.29%	49.1%
2.1	2315/4769	48.54%	104/231	45.02%	51.62%
2.2	2192/4769	45.96%	110/231	47.62%	53.96%
2.3	2113/4769	44.31%	115/231	49.78%	55.44%
2.4	2033/4769	42.63%	117/231	50.65%	57.0%
2.5	1898/4769	39.8%	122/231	52.81%	59.6%
2.6	1733/4769	36.34%	124/231	53.68%	62.86%
2.7	1583/4769	33.19%	131/231	56.71%	65.72%
2.8	1515/4769	31.77%	131/231	56.71%	67.08%
2.9	1410/4769	29.57%	133/231	57.58%	69.14%
3.0	1313/4769	27.53%	133/231	57.58%	71.08%
3.1	1233/4769	25.85%	134/231	58.01%	72.66%
3.2	1164/4769	24.41%	135/231	58.44%	74.02%
3.3	1099/4769	23.04%	135/231	58.44%	75.32%
3.4	1078/4769	22.6%	135/231	58.44%	75.74%
3.5	1013/4769	21.24%	136/231	58.87%	77.02%
3.6	934/4769	19.58%	140/231	60.61%	78.52%
3.7	869/4769	18.22%	140/231	60.61%	79.82%
3.8	832/4769	17.45%	141/231	61.04%	80.54%
3.9	761/4769	15.96%	144/231	62.34%	81.9%
4.0	630/4769	13.21%	144/231	62.34%	84.52%
4.1	557/4769	11.68%	146/231	63.2%	85.94%
4.2	547/4769	11.47%	146/231	63.2%	86.14%
4.3	546/4769	11.45%	146/231	63.2%	86.16%
4.4	482/4769	10.11%	146/231	63.2%	87.44%
4.5	440/4769	9.23%	147/231	63.64%	88.26%
4.6	415/4769	8.7%	147/231	63.64%	88.76%
4.7	320/4769	6.71%	148/231	64.07%	90.64%
4.8	304/4769	6.37%	148/231	64.07%	90.96%
4.9	303/4769	6.35%	148/231	64.07%	90.98%
5.0	303/4769	6.35%	148/231	64.07%	90.98%
5.1	236/4769	4.95%	148/231	64.07%	92.32%
5.2	202/4769	4.24%	148/231	64.07%	93.0%
5.3	202/4769	4.24%	148/231	64.07%	93.0%
5.4	202/4769	4.24%	148/231	64.07%	93.0%
5.5	202/4769	4.24%	148/231	64.07%	93.0%
5.6	184/4769	3.86%	148/231	64.07%	93.36%
5.7	184/4769	3.86%	149/231	64.5%	93.34%
5.8	184/4769	3.86%	149/231	64.5%	93.34%
5.9	184/4769	3.86%	149/231	64.5%	93.34%
6.0	184/4769	3.86%	149/231	64.5%	93.34%

Summary: Shows the data of the bias in increments of 0.1. Shows how the False Positive Percent decreasing as the False Negative Percent increases to get an overall higher accuracy.

Table C.18: Dueling-HMM Bias Result Continuation

Bias	False Positives	False Positive (%)	False Negative	False Negative (%)	Accuracy
6.1	184/4769	3.86%	149/231	64.5%	93.34%
6.2	184/4769	3.86%	149/231	64.5%	93.34%
6.3	184/4769	3.86%	149/231	64.5%	93.34%
6.4	184/4769	3.86%	149/231	64.5%	93.34%
6.5	179/4769	3.75%	149/231	64.5%	93.44%
6.6	178/4769	3.73%	149/231	64.5%	93.46%
6.7	176/4769	3.69%	149/231	64.5%	93.5%
6.8	176/4769	3.69%	149/231	64.5%	93.5%
6.9	174/4769	3.65%	149/231	64.5%	93.54%
7.0	173/4769	3.63%	149/231	64.5%	93.56%
7.1	171/4769	3.59%	149/231	64.5%	93.6%
7.2	166/4769	3.48%	149/231	64.5%	93.7%
7.3	156/4769	3.27%	149/231	64.5%	93.9%
7.4	142/4769	2.98%	149/231	64.5%	94.18%
7.5	140/4769	2.94%	151/231	65.37%	94.18%
7.6	131/4769	2.75%	166/231	71.86%	94.06%
7.7	121/4769	2.54%	168/231	72.73%	94.22%
7.8	117/4769	2.45%	168/231	72.73%	94.3%
7.9	111/4769	2.33%	168/231	72.73%	94.42%
8.0	101/4769	2.12%	169/231	73.16%	94.6%
8.1	97/4769	2.03%	169/231	73.16%	94.68%
8.2	92/4769	1.93%	173/231	74.89%	94.7%
8.3	87/4769	1.82%	175/231	75.76%	94.76%
8.4	78/4769	1.64%	177/231	76.62%	94.9%
8.5	63/4769	1.32%	178/231	77.06%	95.18%
8.6	49/4769	1.03%	183/231	79.22%	95.36%
8.7	45/4769	0.94%	185/231	80.09%	95.4%
8.8	40/4769	0.84%	189/231	81.82%	95.42%
8.9	35/4769	0.73%	190/231	82.25%	95.5%
9.0	33/4769	0.69%	192/231	83.12%	95.5%
9.1	30/4769	0.63%	218/231	94.37%	95.04%
9.2	25/4769	0.52%	218/231	94.37%	95.14%
9.3	16/4769	0.34%	220/231	95.24%	95.28%
9.4	11/4769	0.23%	221/231	95.67%	95.36%
9.5	11/4769	0.23%	222/231	96.1%	95.34%
9.6	10/4769	0.21%	222/231	96.1%	95.36%
9.7	10/4769	0.21%	224/231	96.97%	95.32%
9.8	7/4769	0.15%	225/231	97.4%	95.36%
9.9	6/4769	0.13%	225/231	97.4%	95.38%
10.0	6/4769	0.13%	227/231	98.27%	95.34%
11.0	0/4769	0.0%	230/231	99.57%	95.4%
12.0	0/4769	0.0%	231/231	100.0%	95.38%
13.0	0/4769	0.0%	231/231	100.0%	95.38%
14.0	0/4769	0.0%	231/231	100.0%	95.38%
15.0	0/4769	0.0%	231/231	100.0%	95.38%
16.0	0/4769	0.0%	231/231	100.0%	95.38%
17.0	0/4769	0.0%	231/231	100.0%	95.38%
18.0	0/4769	0.0%	231/231	100.0%	95.38%
19.0	0/4769	0.0%	231/231	100.0%	95.38%
20.0	0/4769	0.0%	231/231	100.0%	95.38%

Summary: Shows the data of the bias in increments of 0.1. Shows how the False Positive Percent decreasing as the False Negative Percent increases to get an overall higher accuracy. Eventually tapering off at around a bias of 10 - 11 where there are no false positives, but misses nearly all the masqueraded data, yet still resulting in an accuracy of 95.38%.