Masquerade Detection via Customized Grammars

Mario Latendresse

Volt Services Group

Science and Technology Advancement Team – FNMOC

U.S. Navy

Overview of the Problem, Context, and Solution

- A masquerade: a computer account is not used by its normal user; this is an intruder
- The observable behavior of the legitimate user and the intruder: executed programs (e.g. the Schonlau datasets)
- Main observation: repeated sequences of programs are often not directly typed by the user
- Generate a grammar to represent the legitimate user and (shared) repeated sequences (the Sequitur algorithm)
- Detecting masquerades based on the grammar
- Experimental results

The Problem & Context

Problem

A (Unix) computer account is used by an intruder, not its normal legitimate user; detect the intruder.

Context

The given behavioral data of the legitimate user: sequences of executed programs.

For Unix: the accounting facility gives the executed programs not the commands directly typed by the user.

This is true for many other behavioral data.

The (Matthias) Schonlau Datasets

Several researchers have used it to benchmark their masquerade detection algorithm.

1. 70 users: 50 users as victims, 20 as intruders
2. 5000 commands for each legitimate user, used as trainig data: automatically generate a (Sequitur) grammar
3. 10000 commands, for each legitimate user, that may be infected by blocks of 100 commands: used to benchmark the detection algorithm

Repeated Sequences

An essential aspect

If scripts are shared between users, there will be common repeated sequences of executed commands in their behavioral data.

Example

Let say `getemail` generates `touch`, `cd`, `pemail`.
The user types: `getemail`, `cd`, `getemail`, `ls`.
Its profile contains:
`touch, cd, pemail`, `cd`, `touch, cd, pemail`, `ls`.

The (local) Profile of the User – A Grammar

These repeated sequences are nested. For example, a script calling another script.

We should extract the nested structure of the repeated sequences and the common repeated sequences between users.

The Sequitur algorithm generates a context-free grammar that detects most of the nested repetitions.

The grammar is generated offline based on a long (e.g. 5000) sequence of commands from the legitimate user.

Sequitur ***Algorithm (1)***

Starting with the main production $S \rightarrow \lambda$ and scanning the string from left to right:

1. add the current command to the end of the right-hand-side of S
2. if the last digram repeats on the right-hand-side of a production, creates a new production for that digram
3. use the new non-terminal instead of the digram: replace the occurrences of the digram by the non-terminal.

Sequitur ***Algorithm (2)***

Generation of Grammar G_1 from `dadabfbfeaeabgbg` **Steps**	**Generation of Grammar G_2 from** `bcabcaca` **Steps**
S $\rightarrow$ dada	S $\rightarrow$ bcabc
S $\rightarrow$ AA A $\rightarrow$ da	S $\rightarrow$ AaA A $\rightarrow$ bc
S $\rightarrow$ AAbfbf	S $\rightarrow$ AaAa
S $\rightarrow$ AABB B $\rightarrow$ bf ...	S $\rightarrow$ BB B $\rightarrow$ Aa ...

Sequitur ***Algorithm (3)***

Generation of Grammar G_1 from `dadabfbfeaeabgbg`	**Generation of Grammar G_2 from** `bcabcaca`
S $\rightarrow$ AABBCCDD A $\rightarrow$ da B $\rightarrow$ bf C $\rightarrow$ ea D $\rightarrow$ bg	S $\rightarrow$ BBC B $\rightarrow$ bC C $\rightarrow$ ca (deleted: A $\rightarrow$ bc)

Production Frequencies and Global Scripts

The user of the grammar is augmented with local and global frequencies on each production.

Production Local Frequency: how many times its expansion occurs in the training data.

Production Global Frequency: how many times its expansion occurs in the other users training data.

A global data base of all productions from all users is created: the global scripts.

Classifying a Block of Commands – Detecting Masquerades (1)

A block of commands is evaluated – if it falls below a threshold (e.g. 30) it is classified as a masquerade.

A block is evaluated by iteratively breaking it into segments:

Positive contribution Each segment matches the expansion of a production of the user grammar. Choose the production that gives the maximum value (use $e(p)$ next slide).

Negative contribution The left over segments, not matching any production expansion, contribute negatively.

Positive contribution – Detecting Masquerades (2)

A production p is valued at:

$$e(p) = l_p \frac{f_p}{f_p + \frac{F_p}{k}} \quad (1)$$

where f_p is the local frequency of production p, F_p its global frequency, l_p the length of its expansion, and k a constant.

Notice that $\frac{f_p}{f_p + \frac{F_p}{k}} \leq 1$.

A good value for k can be found automatically based on the training data. We got $k = 7$, for the Schonlau datasets.

Negative contribution – Detecting Masquerades (3)

Each remaining segment of the block contributes a negative value.

A global script appearing as a substring of an unmatched segment contributes a value of -1 whereas each unseen command contributes a value of -1.

Therefore, the global scripts reduce the negative evaluation.

Reason: the user probably called a new script, not each individual commands.

Updating the Profile

After each testing block – classified as legitimate – the user profile is updated with that block.

The updating applies the Sequitur algorithm on the 100 new commands to the currrent grammar.

There is no updating of the list of common (global) scripts.

Computational Cost

The execution time of Sequitur is linear on the length of the input.

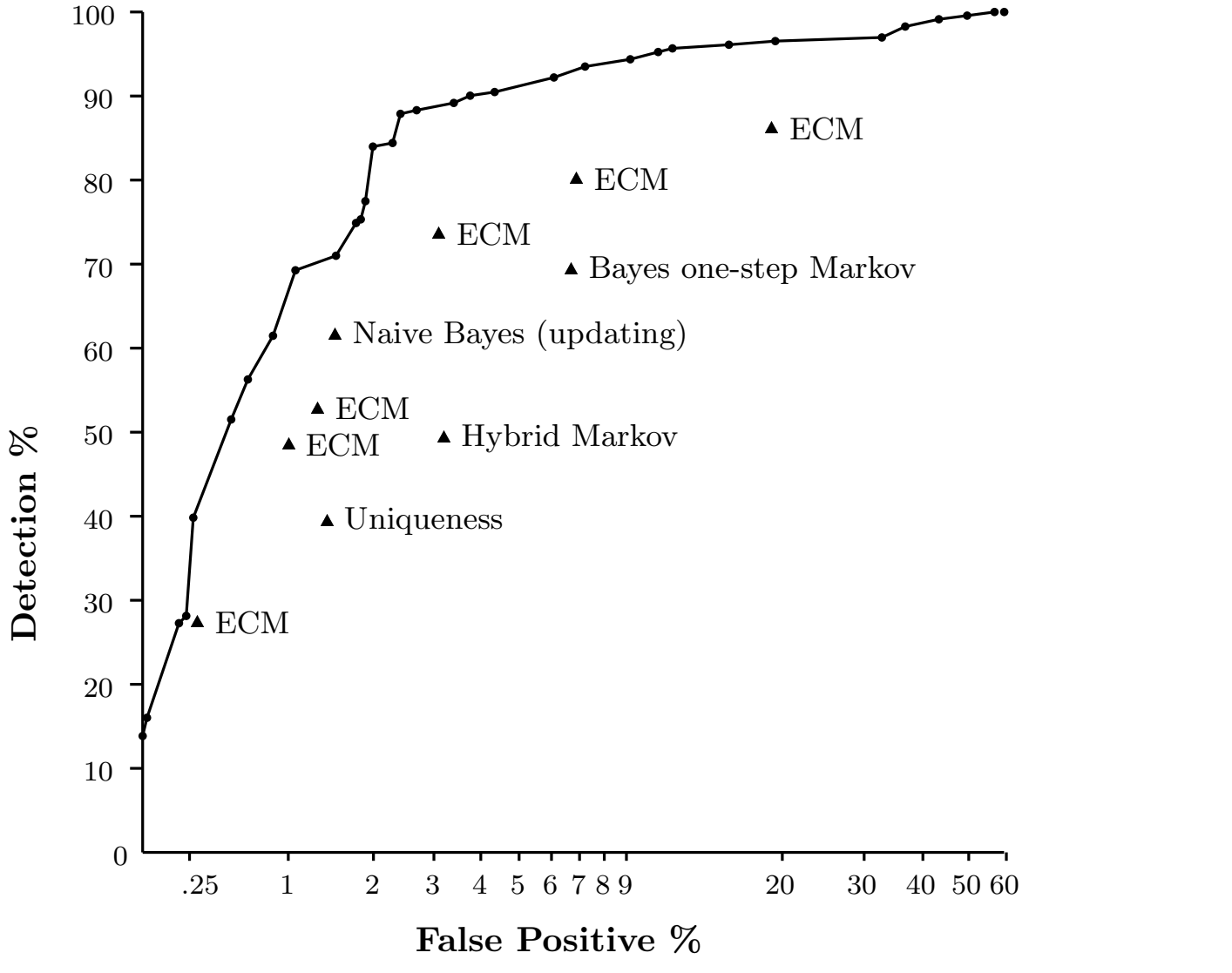
Training

On average, it took less than one second to create the grammar for 5000 commands (2.26GHz, 1GB, Intel Pentium 4).

Detection

On average, verifying one block of 100 commands, including updates, took 127 milliseconds.

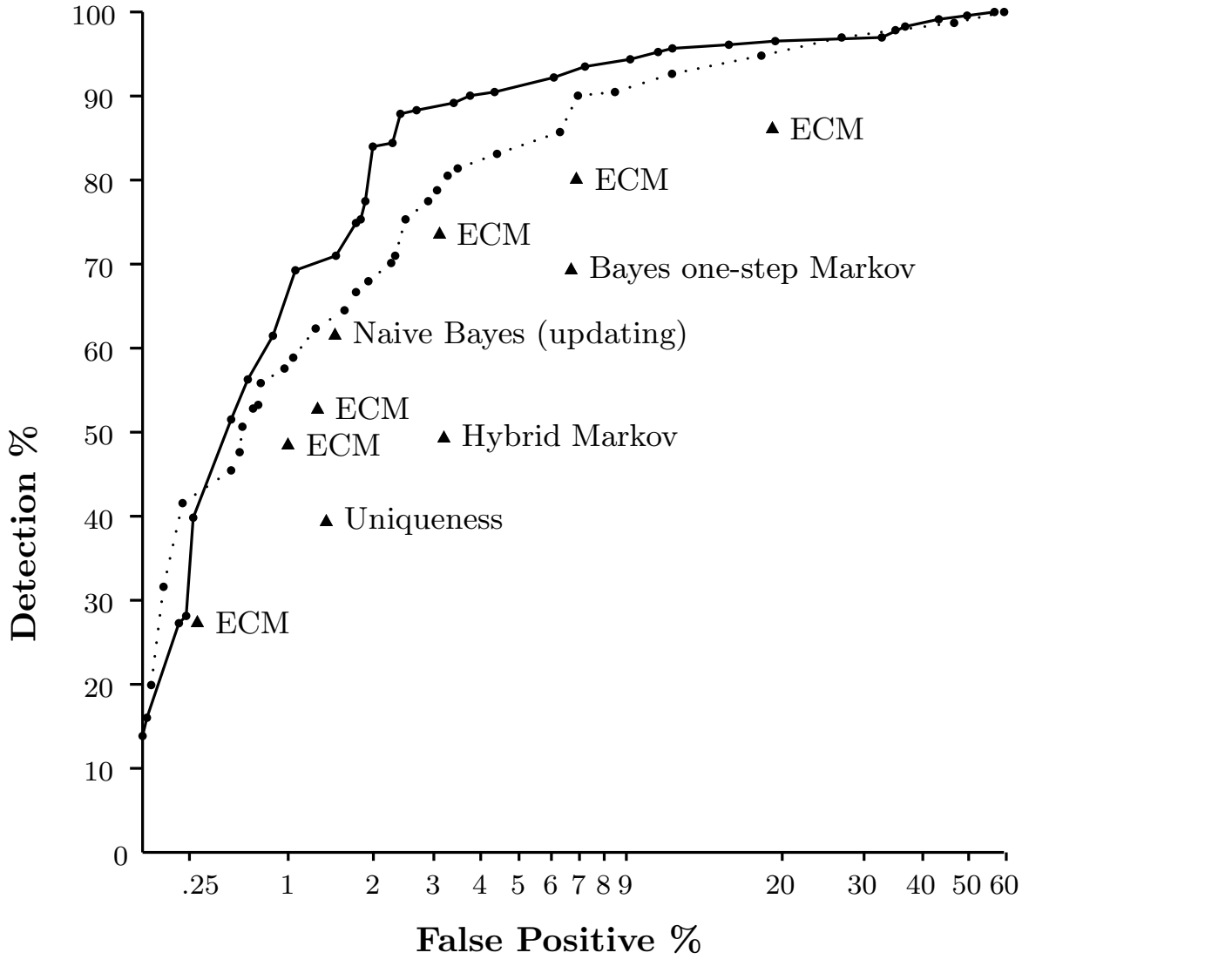
Experimental Results, $k = 7$



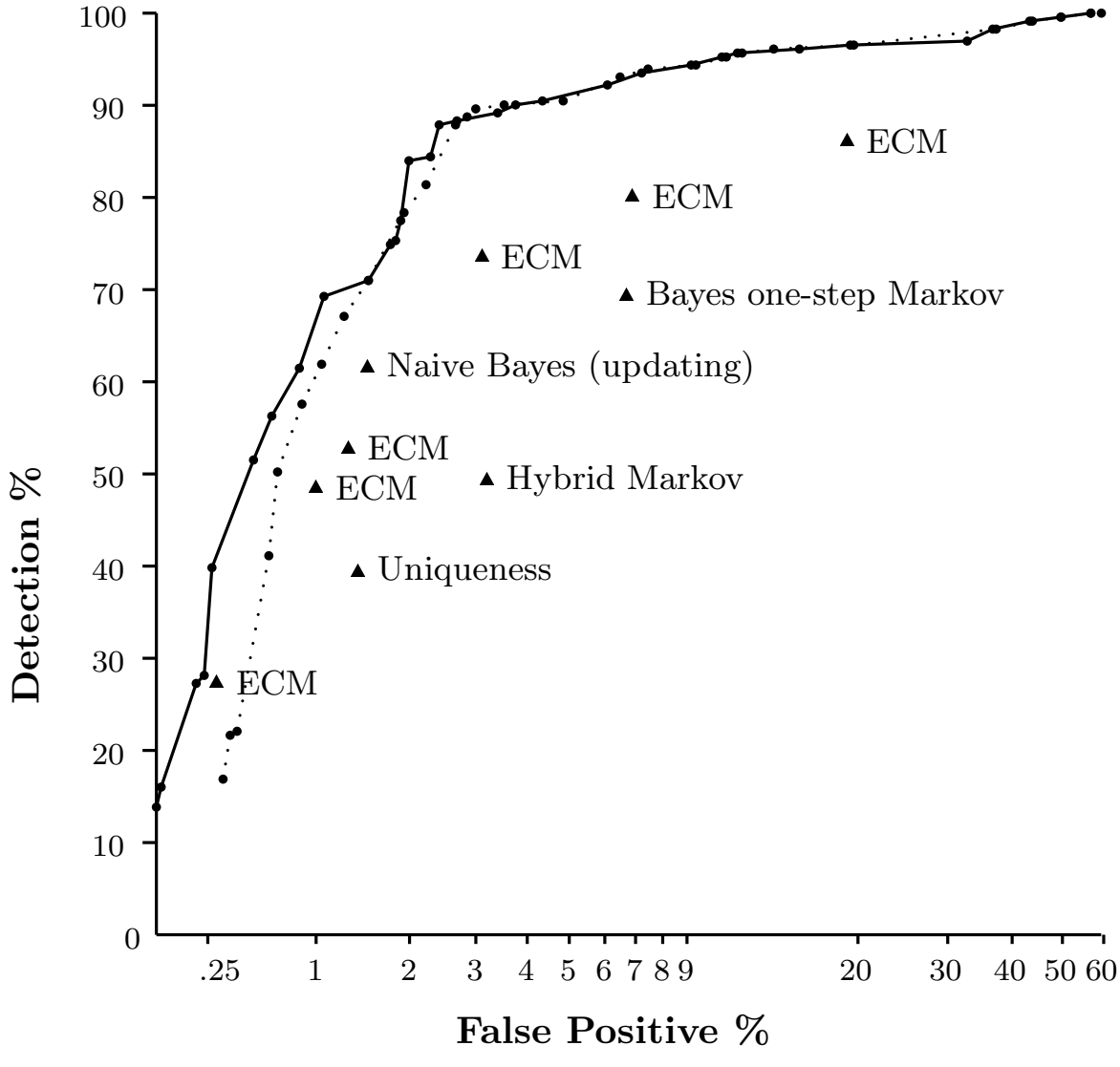
Variations

1. Without the global scripts
2. With $k = 49$ instead of $k = 7$
3. With command frequencies only. This answer the question: are the sequences really useful, perhaps the command frequencies only are as good?

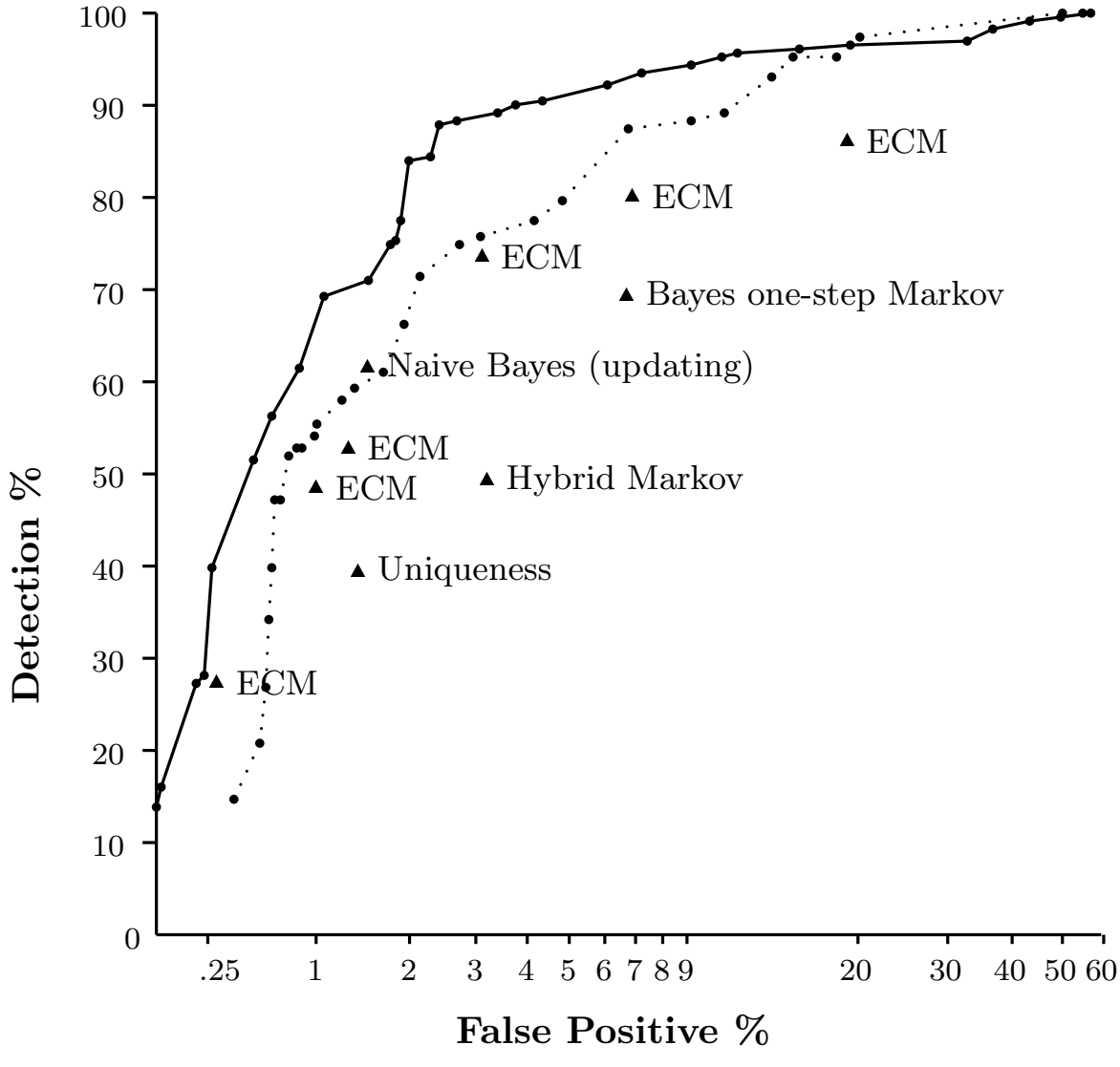
Experimental Results, $k = 49$



Experimental Results, without Global Scripts



Experimental Results, Frequencies Only



Summary

On the Schonlau datasets, our approach gives the highest detection rate across all false positive rates for all known published methods.

Our main improvement: (efficiently) recognising nested (common) repeated sequences from the given behavioral data.

The computational cost is low, it can be applied in real-time.

Could be generalized on more fine grained observables, such as system calls.