Expertise and Behavior of Unix Command Line Users: an Exploratory Study

Mohammad Gharehyazie
Department of Computer Science
University of California, Davis
Davis, CA, USA
Email: gharehyazie@ucdavis.edu

Bo Zhou
Department of Computer Science & Engineering
University of California, Riverside
Riverside, CA, USA
Email: bzhou003@cs.ucr.edu

Iulian Neamtiu
Department of Computer Science
New Jersey Institute of Technology
Newark, NJ, USA
Email: ineamtiu@njit.edu

***Abstract*—Understanding users' behavioral patterns and quantifying users' expertise have a myriad applications, from predicting user actions and tailoring the environment to that specific user, to detecting masquerade attacks and assessing learning outcomes. Toward this end, we have conducted a study on three Unix command datasets, totaling 263 users and more than 1 million commands. We first introduce the notions of command expertise, command line expertise, and command category. Next, we use these metrics, combined with other attributes to define and quantify several user expertise metrics, e.g., category breadth, command line expertise. Our study has revealed many Unix commands characteristics, e.g., Unix command can be grouped into 25 categories; file management is the most common activity; the most commonly used commands are two-characters long. Our study has also revealed many insights into user expertise and behavior, such as: command line length is not an indicator of user expertise; users activity is highest on Monday and decreases every day through Saturday, picking up on Sunday; peak command usage hours are 11 a.m., 1 p.m. and 4 p.m.; development activities happen mostly in the afternoon.**

Keywords–User behavior; user expertise; Unix; empirical study.

I. INTRODUCTION

Unix, Unix-like, and Linux operating systems dominate the market in segments where human-computer interaction is centered around command-line; specifically, the market share of these operating systems, in January 2016, was 66.9% (server), 100% (supercomputer) and 100% (mainframe) [1]. Understanding Unix command usage and the behavior of Unix users has many applications: repetitive sequences of commands can be transformed into scripts for correctness and ease of use; temporal patterns can expose busy and idle periods hence allowing capacity to be scaled accordingly; and noticing that a user U's commands or temporal access patterns are markedly different compared to U's prior commands and usage patterns can indicate a masquerade attack [2] (i.e., U's account has been compromised and is being used by a malicious user M). Quantitative measures of user expertise can serve as a base for comparing users (e.g., who are the most competent or efficient users?), inferring user roles (e.g., student vs. seasoned developer vs. system administrator) or assessing how users learn.

Few studies have focused on understanding user behavior based on command line usage. Rather, prior studies' focus has been on: masquerade detection (malicious users taking control of a legitimate user's account) [3][4][5][6][7][8][9]; user session characterization [10] in terms of commands per session, errors encountered and directories explored; or predicting high-level user actions [11]. In Section II, we provide a detailed comparison with related work. However, none of these previous studies have attempted to quantify user or command expertise and study temporal patterns associated with users/expertise.

We start by describing the datasets and the approach we have used to identify commands and their arguments (Section III). Identifying individual commands is nontrivial due to several reasons, such as the myriad ways in which Unix commands can be chained or used as arguments for other commands.

In Section IV, we discuss the methodology we have used for quantifying expertise and assigning expertise values to commands and users. First, we assign an *expertise* value to each command — the higher the value, the higher the probability that the command is used by more advanced users, or requires more advanced knowledge. Next, using command expertise, we assign expertise values to entire command lines. We then group commands into categories such as file management, editor, and compiler. Based on these metrics, we introduce metrics for users expertise: *user category breadth* to quantify user expertise in terms of the number of different categories employed by that user, as well separating users into high- and low-expertise groups.

In Section V, we present our findings. We first characterize commands in each dataset; we found that file management is the principal activity, with `cd` and `ls` accounting for substantial percentages of user commands. We found that the most common command length is *two*; that command line length is not necessarily an indicator of expertise, and that commands/users permit a natural separation into high- and low-expertise commands/users.

We also analyzed temporal patterns in one of the datasets (the only dataset to contain command timestamps). We found that command activity tends to decrease from Monday until Saturday, and increase slightly on Sunday; that peak command usage hours are 11 a.m., 1 p.m. and 4 p.m.; and that development activities (use of editors and compilers) peaks in the afternoon.

II. RELATED WORK

Schonlau et al. [3] have used various statistical methods to detect masquerade attacks by finding "bad data" (attacker's

TABLE I. FEATURE AVAILABILITY FOR EACH DATASET.

datasets	Mahajan	Greenberg	Schonlau
Users	45	168	50
Commands per user	709*	1,441*	15,000
Command timestamp	✓		
Session start/end		✓	
Command arguments	✓	✓	
Command chaining	✓	✓	
User Group		✓	

* Median number of commands per user.

commands) inside sequences of "good data" (benign commands issued by a legitimate user). Other studies have similarly used machine learning or grammars for masquerade detection [4][5][6][7][8][9]. Greenberg [10] has collected traces of 168 users via a modified `csh` shell; they partitioned the users into four non-overlapping categories (novice programmers, experienced programmers, computer scientists and non-programmers) and characterized user sessions in terms of commands per session, errors encountered and directories explored. Chinchani et al. [12] has focused on the reverse problem of generating user models so that synthetic command traces can be generated automatically. Fitchett and Cockburn [11] developed a predictor model for revisitation/reuse based on user actions (commands, window switching, URL accesses).

Note that our focus is different compared with the aforementioned efforts. Rather than finding suspicious/anomalous commands for purposes of masquerade detection, we aim to answer more general questions: how can command expertise, command line expertise, and user expertise be operationalized? How can users and commands be grouped into categories that generalize and are stable across datasets? What are the temporal patterns associated with commands, command categories, and users, e.g., when (time-of-day/day-of-week) does development happen, as opposed to editing, and when are experts active compared to novice users?

III. DATASETS

Our analysis is based upon three main datasets — collected by other researchers [9][3][10] — totaling 263 users and 1,023,993 commands. Table I provides an overview of the datasets and the features they include: each dataset consists of real commands collected from actual usage — ranging from 709 up to 15,000 commands per user. The dataset attributes vary: while Mahajan's set contains a timestamp for each command, the other sets do not; conversely, Greenberg's set has session start and end markers while the other two do not. Finally, Mahajan and Greenberg's sets contains command arguments, including the chaining of multiple commands on the same line (e.g., via the pipe operator), while Schonlau's only contains a command prefix (first 8 characters).

We now describe each dataset, its features, and the methodology we used to extract characteristics from that particular set.

1) Mahajan: The richest, most detailed data was gathered by Mahajan [9]: command traces for 45 users. Each command has a timestamp and the entire command line, e.g., including complex pipes, was captured. To parse each line, we first separate the command portion into sections of commands by pipes (the '|' character) and logical commands such as '&&'. After separating a line into sections, each section contains a command, along with zero or more parameters. For example, the following line will be separated into 3 sections:

```
ls -l | grep key | less
```

Backquotes or "backticks" are commands that are executed before the rest of a line, and their result is "pasted" at their position in the line. For example:

```
vim notes.`date +%F`
```

Hence we look for backquotes in each section and treat it as a separate section; we find the command and its parameters (as described below) and we treat the whole section as an extra parameter for the section which contained the backquotes.

To count the number of parameters for each command, we separate each section by space and redirection characters ('>', '>>', '<', and '<<'). As mentioned above, each section is split based on space or redirection characters. Redirection to/from a file is also considered a parameter. In some rare cases there are two commands per section such as `sudo apt-get install blah` which contains two commands ('`sudo`' with 0 parameters and '`apt-get`' with 2 parameters). Only two commands were found that used such a feature: '`sudo`' and '`time`'.

To conclude, in the aforementioned example, we detect 3 commands: '`ls`' with 1 parameter, '`grep`' with 1 parameter, and '`less`' with no parameter. Notice that all commands after the first pipe have one more parameter than immediately visible, and that is because the other parameter has been piped.

Finally, we proceeded to identifying *sessions*, i.e., start and end of time intervals when users started the command line interaction. While Mahajan's dataset is very rich in most aspects, it does not explicitly record session information, so to identify sessions we computed the time intervals between consecutive commands, plotted their distribution, and visually identified cut-off points hence session begin/end.

2) Greenberg: This dataset consists of 168 Unix users [10]. Like Mahajan's dataset, it contains command parameters, and complex command lines, but it does not contain timestamps like Mahajan's. It does however contain session delimiters. Another feature of this dataset is that it categorizes users into 4 categories based on their expertise. In detail, there are 52 computer scientists, 36 experienced programmers, 55 novice programmers and 25 non-programmers respectively. This makes it particularly valuable in evaluating our expertise extraction methods. Next, we show an example of the information contained in Greenberg's dataset (the dataset contains more information which is irrelevant to this study and has been removed from the example for clarity).

```
S Wed Feb 18 16:37:25 1987
E Wed Feb 18 16:56:22 1987

C date
C nroff terry.abs | enscript
C p audio.mail

S Fri Feb 20 12:41:39 1987
E Fri Feb 20 14:24:06 1987

C mail
```

```
C rlogin sun-fsa
C rlogin sun-e
C rlogin sun-b
...
```

3) Schonlau: Although this dataset is large, containing 15,000 command traces per user for 50 users, it is limited in three ways. First, the parameters for each command are omitted; this leads to missing complex pipes and chains of commands, which can be used to assign user expertise. Second, due to the method used for gathering data, system commands, e.g., commands invoked from C programs via `system()` or `exec()` were included. Finally, the command traces do not have timestamps, so processing any temporal information such as number and duration of each user session is impossible. On the flip side, parsing this dataset is trivial, since each line contains a simple command, with no parameters, no loops, no pipes, etc.

IV. EXPERTISE

We used the following approach for developing a uniform notion of expertise across the three datasets: we first assign an expertise value $E(C)$ to each command C, then for each command line L, including lines that consist of the chainings commands $C_1, \ldots, C_n$, we assign an expertise coefficient $lCoef(L)$ based on constituent command(s) expertise as well as the command chaining information.

Command Categories. We group commands into *categories*, e.g., `vim` or `emacs` are categorized as *editor* commands, while `ping` or `traceroute` are *network* commands, and so on. In total we have defined 25 categories. The number of categories varies little across datasets: 24 for Mahajan, 23 for Schonlau and 18 for Greenberg (Table II) which indicates that our category definitions are stable.

A. Command and Line Expertise

1) Assigning Expertise to Commands: Due to inherent differences among the three datasets, they could not be merged into one dataset, and thus assigning expertise was performed separately on each of them. We first measure the number of users that have used each command (U), along with the number of times each command was used ($Freq$). We also manually assign commands to categories based on type of application.

We group U into 4 bins and $Freq$ into 3 and assign an expertise value to each bin (U_{exp} and $Freq_{exp}$) as shown on the left of Figure 1. A level of expertise is assigned to each category (C_{exp}) as shown on the bottom of Figure 1, e.g., browser commands have expertise value 4, network and svn (version control) have value 10, while compiler commands have expertise value 14. Then, for each command we assign an intra-category expertise adjustment (I_{exp}), as explained next.[1] The reasoning behind the expertise assignments is:

- For user breadth U_{exp}, the more people utilizing a command, the less likely it is to require high expertise. At the same time, if a command is used only by a single person, there is a high chance that it is highly personal, e.g., scripts. We believe the expertise associated with this bin should be lowered, but not lower than the bin with highest number of users.
- For $Freq_{exp}$, a command that is used less frequently is likely associated with higher expertise.
- A category of commands C_{exp} is the highest indicator of expertise, e.g., a compiler command is much more complicated than a browser command.
- Within a category, there are certain commands that require a higher expertise to be used; these commands are given extra expertise points in intra-category expertise I_{exp}.

[1]Expertise values were assigned based on consensus among authors.

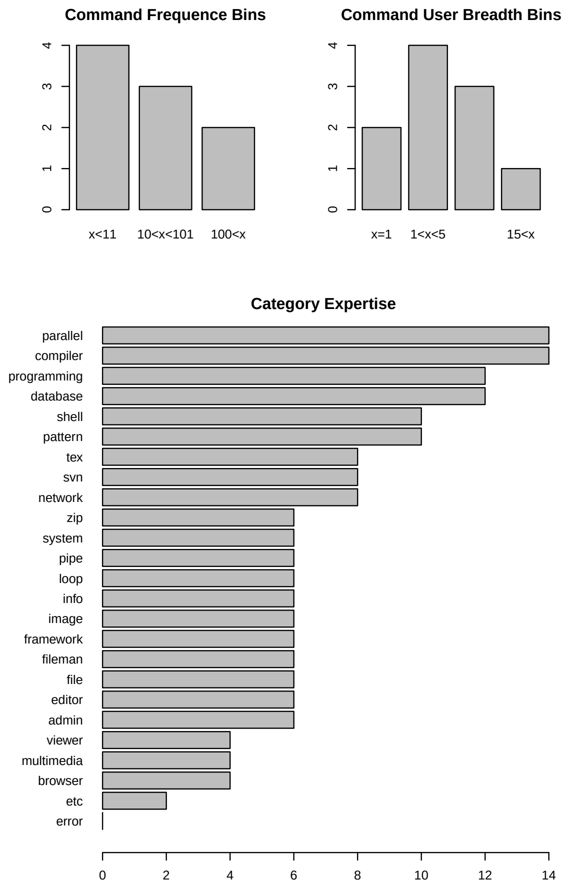

Figure 1. Expertise assigned to bins of a) command frequency (top left); b) user breadth (top right); and c) command category (bottom) for the Mahajan dataset.

Therefore, we use the following formula for assigning expertise to each command C_i:

$$Exp(C_i) = U_{exp}(C_i) + Freq_{exp}(C_i) + C_{exp}(C_i) + I_{exp}(C_i) \quad (1)$$

While the most influential element of each command's expertise is the category it belongs to, other elements can boost or hinder its expertise and thus provide a wide range of values. For example, in Schonlau's dataset, expertise values range from 6 to 22.

2) Assigning Expertise to Lines: For the dataset of Schonlau et al., each line's expertise is the same as the command in that line. But for Mahajan and Greenberg datasets, we need a measure to combine multiple commands' expertise into a single value. Adding up all the expertise is not a great idea since it can highly inflate the expertise values, and it is certainly misleading, since someone who uses 10 commands in a single line, does not necessarily possess 10x greater expertise in comparison to another person who uses the same commands on separate consecutive lines. We believe that the former's expertise is merely a fraction more than the latter's and that is due to their ability to combine multiple commands in a single line and perform complicated tasks. To this end, we have developed the following line expertise computation scheme:

$$lCoef(l_i) = \begin{cases} 1 + \frac{min(|Commands(l_i)|,4)}{10}, & \text{if } l_i \text{ contains backquotes} \\ 1 + \frac{min(|Commands(l_i)|,4)-1}{10}, & \text{otherwise} \end{cases} \quad (2)$$

$$Coef(C_j) = 1 + \frac{min(|Params(C_j)|,4)-1}{10} \quad (3)$$

$$Expertise(l_i) = lCoef(l_i) \times \max_{C_j \in Commands(l_i)} Exp(C_j) \times Coef(C_j) \quad (4)$$

Here, l_i is any given line, C_j is a member of the set of commands within l_i, denoted by $Commands(l_i)$ and $Params(C_j)$ gives the set of parameters for C_j. $lCoef(l_i)$ is an expertise coefficient for line l_i which gives a 10% boost to l_i's expertise for each extra command in l_i, up to 4 commands; it also gives an extra 10% boost if backquotes are used in l_i. $Coef(C_j)$ is a command coefficient which gives a 10% boost to C_j's expertise for each extra parameter provided to C_j, up to 4 parameters.

3) High- and Low-Expertise Commands: To provide a straightforward binary separation into "easy" and "advanced" commands, we use the command expertise distributions to divide all commands into low and high expertise groups. These in turn will be used to support separating users into low- and high-expertise groups. We present the results in Section V-A3.

B. User Expertise

We now provide definitions of user expertise.

1) User Category Breadth: User Category Breadth (UCB), i.e., the number of different categories they actively use, is also an indicator of expertise. To measure UCB, we identify the category of commands for each user, and apply a minimum number of commands threshold to weed out those cases where accidental or extremely infrequent use of commands would count toward category use. Specifically, we used 1.5% of the average number of commands for each user in each dataset to define the aforementioned threshold — this translates to 20 commands in Mahajan's dataset and 250 commands in Schonlau's, i.e., to count a category C toward a user U's UCB, U has to have used at least 20 and 250 commands in C, respectively.

2) High- and Low-Expertise Users: To distinguish different users, we divide them into low and high expertise groups based on high- and low- command line expertise mentioned in Equation (2). We found a clear delineation between the high- and low- expertise sets; we present the results in Section V-B2.

V. Results and Discussion

We now proceed to discuss our findings.

TABLE II. SUMMARY OF COMMAND CHARACTERISTICS.

	Mahajan	Greenberg	Schonlau
Total Commands	56,261	313,169	750,000
Unique Commands	1,218	4,117	856
Categories	24	18	23

TABLE III. TOP-20 COMMANDS FOR EACH DATASET.

Rank	Mahajan		Greenberg		Schonlau	
	Command	%	Command	%	Command	%
1	`cd`	15.3	`ls`	12.3	`sh`	8.7
2	`ls`	15.0	`cd`	8.8	`cat`	4.3
3	`git`	3.7	`pix`	6.2	`netscape`	4.3
4	`vim`	3.3	`umacs`	4.9	`generic`	4.1
5	`sudo`	3.1	`e`	4.2	`ls`	4.0
6	`grep`	2.8	`rm`	3.1	`popper`	3.3
7	`vi`	2.6	`fg`	3.1	`sendmail`	2.8
8	`gvim`	2.3	`emacs`	3.0	`date`	2.7
9	`ssh`	2.2	`more`	2.8	`rm`	2.3
10	`mm`	2.2	`lpq`	1.9	`sed`	2.1
11	`rm`	2.0	`mail`	1.8	`nawk`	2.0
12	`java`	1.9	`lpr`	1.8	`expr`	1.9
13	`perl`	1.6	`cat`	1.8	`tcsh`	1.8
14	`javac`	1.5	`cp`	1.4	`grep`	1.7
15	`find`	1.4	`ps`	1.3	`tcpostio`	1.4
16	`clear`	1.2	`nroff`	1.2	`uname`	1.4
17	`mount`	1.1	`who`	1.1	`ln`	1.3
18	`cp`	1.1	`make`	1.0	`hostname`	1.3
19	`cat`	0.9	`fred`	0.9	`gcc`	1.3
20	`exit`	0.8	`u`	0.8	`true`	1.3

A. Command Characteristics

1) Command Distribution: Table II shows the summary of each dataset. Although the Schonlau dataset has the most commands, it has the fewest *unique* commands. Note how, despite differences in datasets (e.g., provenance, year of collection, Unix system used) they have similar numbers of command categories, which indicates that our category definitions are quite stable across different Unix user populations.

Table III shows the top-20 most used commands and their percentage in each dataset. File management commands (`ls`, `cd`, `cp`, and `rm`) are the most popular by far, and they dominate the Mahajan and Greenberg datasets (more than 20% of their commands fall into this category). The Schonlau dataset is more evenly distributed, but file management is popular there as well.

Furthermore, we investigate whether commands differ between user groups. Table IV shows the top-20 most used commands and their percentage of user groups in the Greenberg dataset. For groups computer scientist, experienced programmer and non programmer, similar to the whole dataset, file management commands contribute the most, but for the novice programmer group, the compiler-related command `pix` and (Pascal interpreter and executor) and editor command `umacs` are the most used commands.

Table V shows top-5 most used categories and their percentage in each dataset. The observations are similar to the previous observations on commands: file management is prevalent, with Mahajan and Greenberg's datasets having a higher concentration of such commands compared to Schonlau's.

The Greenberg dataset comes with an assignment of users into groups: computer scientists, experienced programmers, novice programmers, and non-programmers [10]. Therefore,

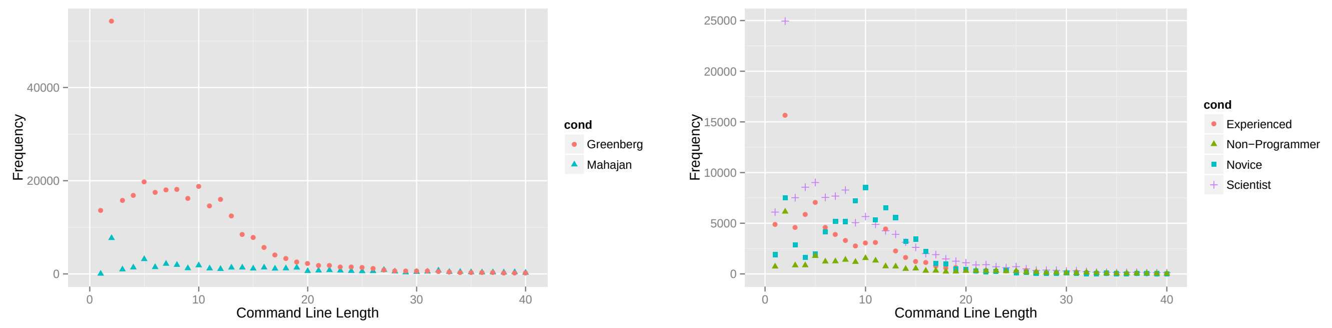

Figure 3. Distribution of command line length in Mahajan and Greenberg datasets (left) and the four user groups of the Greenberg dataset (right).

TABLE IV. TOP-20 COMMANDS OF EACH CATEGORY IN GREENBERG DATASET.

Rank	Scientist		Experienced		Novice		Non-Programmer	
	Command	%	Command	%	Command	%	Command	%
1	`ls`	14.6	`cd`	12.0	`pix`	24.4	`ls`	15.8
2	`cd`	10.5	`ls`	11.8	`umacs`	19.7	`emacs`	10.6
3	`e`	5.2	`e`	5.8	`ls`	7.8	`nroff`	9.1
4	`fg`	4.4	`fg`	4.6	`rm`	3.4	`cd`	8.5
5	`rm`	3.0	`more`	4.1	`u`	3.1	`e`	5.3
6	`mail`	2.8	`rm`	2.7	`cd`	2.9	`rm`	4.0
7	`emacs`	2.4	`make`	2.7	`cat`	2.7	`ee`	3.8
8	`lpq`	2.2	`emacs`	2.5	`more`	2.6	`more`	3.4
9	`more`	2.1	`lpr`	1.9	`script`	2.4	`lpr`	3.0
10	`ps`	1.8	`l`	1.9	`lpr`	2.4	`hpr`	2.8
11	`f`	1.6	`cat`	1.8	`lpq`	2.0	`ptroff`	2.2
12	`cat`	1.6	`ada`	1.8	`cp`	2.0	`lpq`	1.9
13	`who`	1.5	`examples_vax`	1.7	`emacs`	1.8	`ps`	1.8
14	`mv`	1.1	`cp`	1.5	`pi`	1.5	`cp`	1.4
15	`lpr`	1.1	`a.out`	1.3	`p`	1.2	`tbl`	1.4
16	`man`	1.1	`rwho`	1.3	`mail`	1.0	`w`	1.3
17	`rlogin`	1.0	`mail`	1.2	`fred`	1.0	`col`	1.2
18	`cp`	1.0	`lpq`	1.2	`logout`	0.8	`mail`	1.2
19	`page`	0.9	`bye`	1.2	`pdpas`	0.7	`rr`	1.1
20	`fred`	0.9	`ps`	1.2	`man`	0.6	`spell`	1.1

TABLE V. TOP-5 CATEGORIES FOR EACH DATASET.

Rank	Mahajan		Greenberg		Schonlau	
	Category	%	Category	%	Category	%
1	fileman	39.0	fileman	28.3	pattern	13.7
2	etc.	8.4	editor	14.7	framework	12.2
3	editor	8.2	info	13.1	etc.	12.1
4	pattern	5.8	etc.	10.7	system	11.8
5	compiler	5.1	compiler	9.2	fileman	10.1

TABLE VI. TOP-5 CATEGORIES FOR USER GROUPS OF THE GREENBERG DATASET.

Rank	Scientist		Experienced		Novice		Non-Programmer	
	Category	%	Category	%	Category	%	Category	%
1	fileman	32.3	fileman	31.7	compiler	27.4	fileman	31.2
2	info	14.4	etc.	13.2	editor	23.3	editor	21.1
3	etc.	11.8	info	12.6	fileman	17.2	info	13.1
4	editor	10.6	editor	10.7	info	11.6	Tex	11.8
5	system	9.1	system	8.4	system	7.4	etc.	11.1

for this dataset alone, we have investigated category distribution for each group. According to Table VI, we found that computer scientists and experienced programmers' groups have similar top categories, e.g., file management is the most frequent used commands. However, the novice programmer group used compiler and editor commands most often, whereas non-programmers, as expected, used file management, editor, help (info) and TeX (text processing) commands most often.

2) Command Line Expertise: We now illustrate how command line expertise differs among user groups. We classified the Mahajan dataset into 3 groups: experienced programmer, novice programmer and non programmer; we performed this classification manually by sampling each user's commands and evaluating the categories and expertise of the sampled commands. Figure 2 shows the bean-plot of average command line expertise of each group in Mahajan. The shape of the bean-plot is the entire density distribution, the short horizontal lines represent each data point, the longer thick lines are the medians, and the white diamond points are the means. While the results show that the peak density of expertise is higher for the experienced users and then for novice programmers, a *Mann-Whitney U test* shows that the differences are not statistically significant, which very well may be due to our small sample size (a total of 45 users).

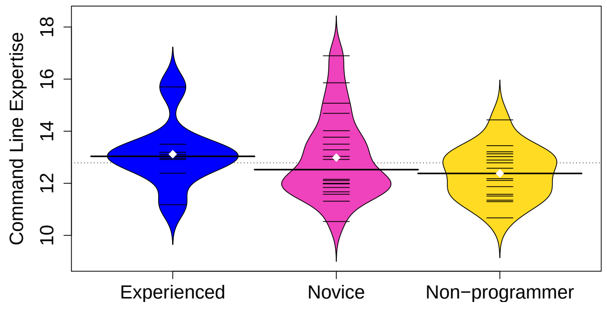

Figure 2. Beanplot of average command line expertise of each user group in Mahajan's dataset.

We use command line length as a metric to check the difference between the Mahajan and Greenberg datasets, as well as the difference between user groups in Greenberg. According to the left side of Figure 3, command line length 2 is the most frequent pattern for both Mahajan and Greenberg which is expected, as commands `ls` and `cd` have been used most often. For scientist and experienced programmers of Greenberg dataset, we found a similar pattern. But for novice programmers, command line length 10 is the most used, while command line length 9 has similar frequency with command line length 2. Upon investigation, we found that since novice programmers were learning how to program, they used commands `umacs` and `pix` (with corresponding arguments to reach line lengths 9 and 10) more often, which result in the different trend compared to scientists and experienced programmers.

3) High- and Low-Expertise Commands: Finally, we investigate the distribution of expertise for each dataset; in Figure 4, we show the result. We set the median of expertise values (which empirically is 11 across all three datasets) as the

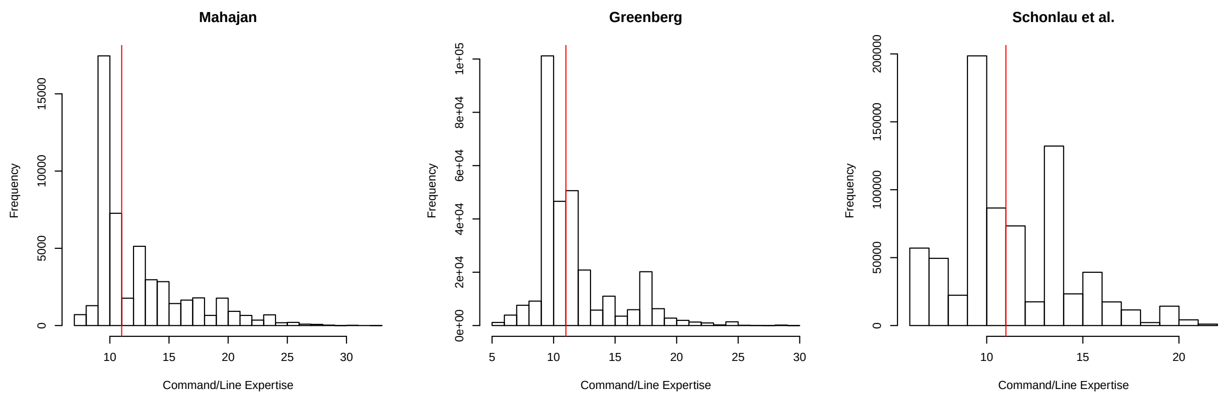

Figure 4. Distribution of expertise in commands. The red line indicates the median, which separates *low* and *high* expertise commands.

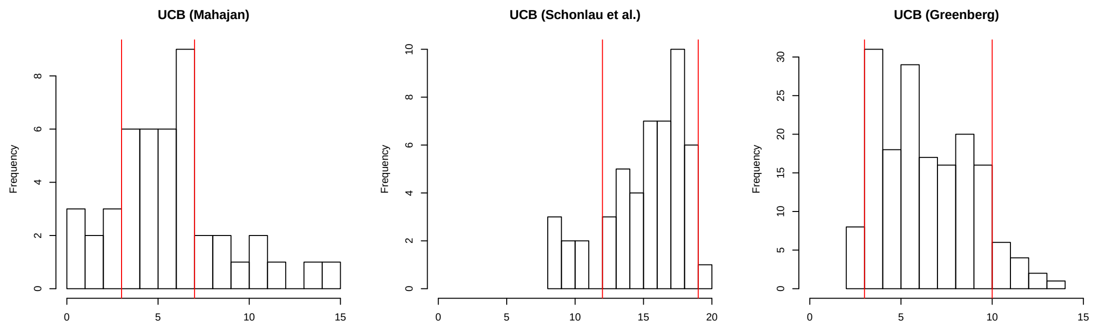

Figure 5. Histogram of number of categories of commands each user is active on. Being active in a category requires using one or more commands from that category at least 20 times.

threshold to separate *low* from *high* command line expertise. Note that, since the same value, 11, emerges as a natural threshold in all three distributions, we gain confidence in the stability of our command expertise metrics.

B. *User Characteristics*

1) User Category Breadth: We analyzed Mahajan's dataset as described above for UCB classes. The results are shown in Figure 5 (left). We observe 3 major patterns. First, there is a group of people who use 3 or fewer categories of commands. Second, there is a large group of people who use between 4 and 8 categories, and finally the group of people who use more than 8 categories. We name these classes *low*, *medium*, and *high* breadth classes respectively. In Mahajan's dataset, we found that 6 users are a member of low-breadth class, 10 are a member of high-breadth class, which leave the rest (29 users) in the medium-breadth class.

For Schonlau's dataset the pattern occurs again, but at different points. Here the *low* breadth class spans up to 12 categories and consists of 7 users. The *high* breadth class starts from 19 categories and consists of 1 user. This leave the *medium* group with 42 people and a range of 12 to 19 categories.

For the dataset of Greenberg, we still have similar pattern with different points. The *low* breadth class spans up to 3 categories and consists of 8 users. The *high* breadth class starts from 10 categories and consists of 29 users. Then the *medium* breadth class group with 131 users and a range between 3 to

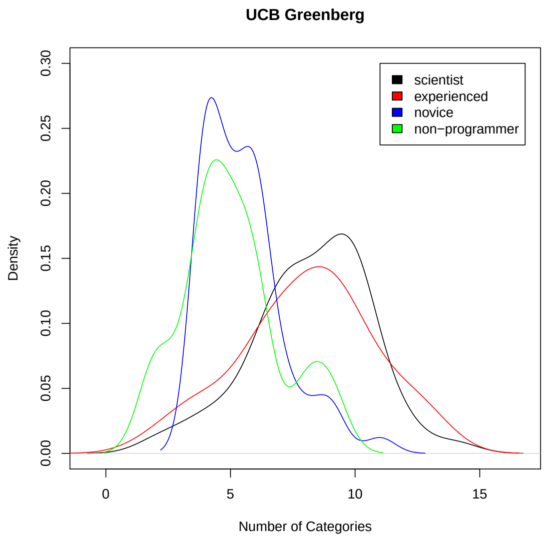

Figure 6. Kernel density function for the command category distribution of each user class in Greenberg's dataset.

10 categories. While the values are different, the pattern of two small classes with low and high breadth and a larger class of medium breadth is still prevalent.

We also analyzed category usage in the different user groups of Greenberg's dataset. Figure 6 shows the kernel density function for the command category distribution of each user group. We found that novice programmers and

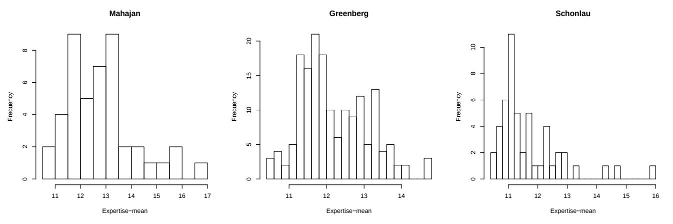

Figure 7. Histogram of mean command line expertise of each user is active on.

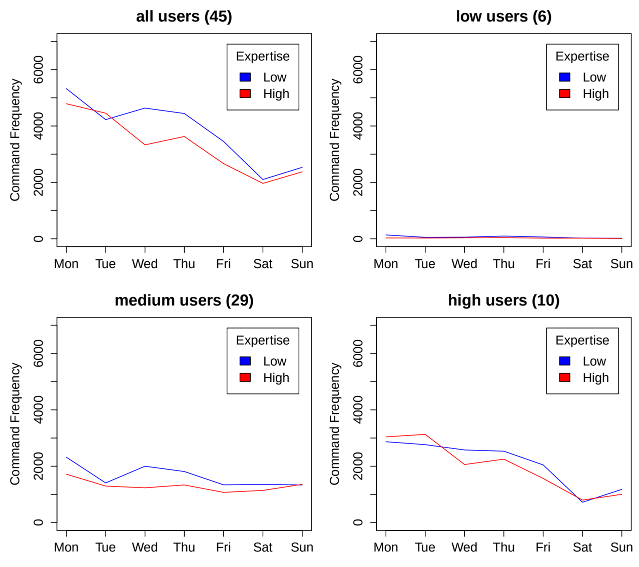

Figure 8. Frequency of low and high expertise commands used by different classes of users at different days of the week.

non programmers employ fewer command categories than scientists and experienced programmers, which is intuitive.

2) High- and Low-Expertise: We now show the results of command line expertise per user. Figure 7 shows the result. Greenberg and Schonlau datasets have similar trends, i.e., most users have only one peak value between 11 and 12; for the Mahajan dataset, we found similar trends, i.e., two peak values between 11 and 13. We believe the similarities across datasets validate our choice of expertise metrics.

C. Day-of-week Command Patterns

To understand temporal patterns for command usage, we studied Mahajan's dataset (the only dataset that come with timestamps). This process is performed for all users, and then for each breadth category separately.

Figure 8 shows the distribution of command usage over days of week. The red lines depict high expertise commands' frequency while the blue lines indicated low expertise commands' frequency. The week starts on Monday. While the frequency tends to decrease from Monday through Sunday, we need to split users into expertise levels to better understand trends. Interestingly: (a) low- and medium-expertise users show little variation (slight decrease) as the week progresses, while for high-expertise users the trend is clear and decreasing; (b) Sunday usage is higher than Saturday usage.

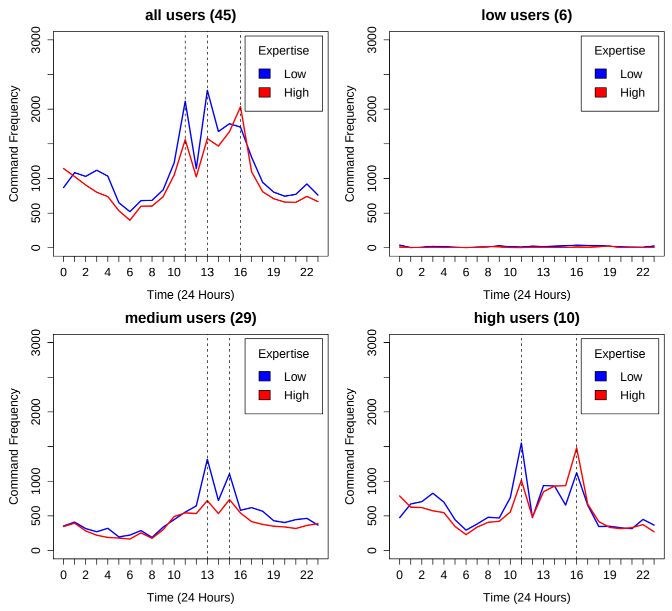

Figure 9. Frequency of low and high expertise commands used by different classes of users at different hours of the day.

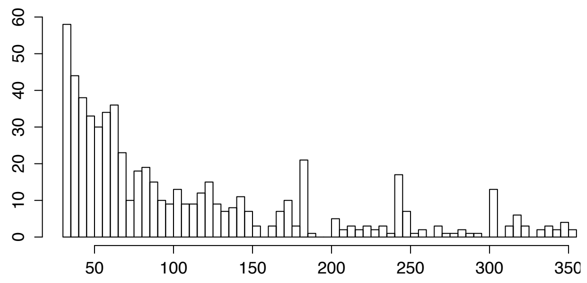

Figure 10. Histogram of temporal distance between consecutive commands for high-expertise users in the Mahajan dataset: x-axis is intercommand time in minutes, y-axis is frequency.

D. Time-of-day Command Patterns

The distribution of command usage over hours of day is shown in Figure 9. The red lines depict high expertise commands' frequency while the blue lines indicated low expertise commands' frequency. There is a clear spike in activity around 11 a.m. and a rise in activity from 1 p.m. to 4 p.m. But when we split the users based on UCB, we observe different patterns for medium and high breadth classes. The Medium breadth class shows spikes of activity around 1 p.m and 3 p.m., while the high breadth class shows a spike around 11 a.m., and a relatively high activity from 1 p.m. that peaks at 4 p.m.

E. Inter-command Time

We found that users interact with the shell in sessions, i.e., bursts of commands coming in rapid sequence, followed by long pauses. We studied inter-command time to get a sense as to how temporally close the commands are. Figure 10 illustrates this for high-expertise users in the Mahajan dataset. Note that the distribution of inter-command distances is highly skewed towards zero and has a very long tail (we trimmed both ends of the distribution — less than 30 minutes and more than 6 hours — in order to have a clearer view). As we can see in

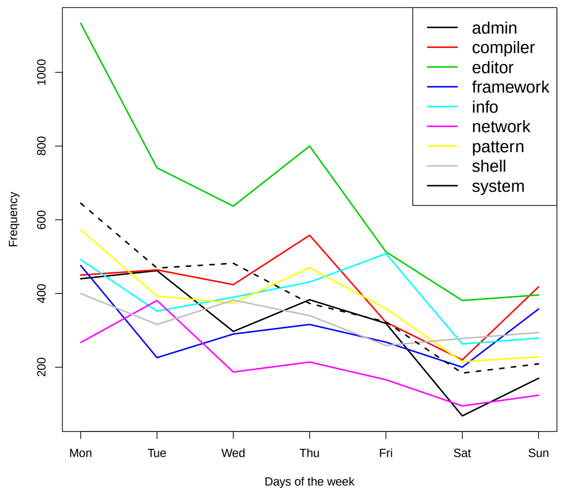

Figure 11. Frequency of several command categories used by all the users at different days of the week.

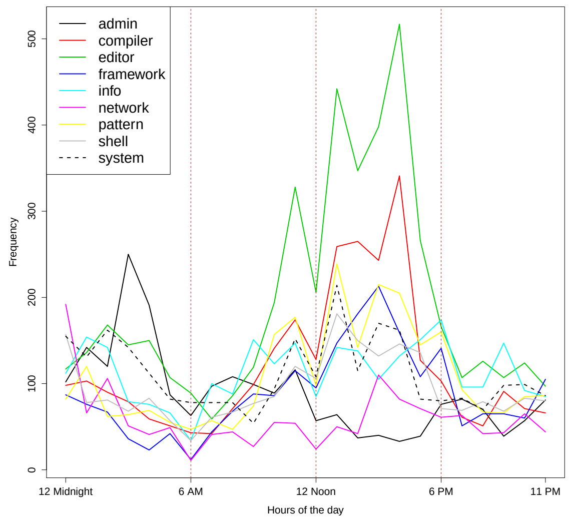

Figure 12. Frequency of several command categories used by all the users at different hours of the day.

the figure, the distribution within the defined range still looks mostly exponential.

F. Category Patterns

We also analyzed the usage pattern of categories. Figure 11 shows category frequency for each day of week: Monday and Thursday are the days with the highest activity. Moreover, "editor" is the most popular category.

Figure 12 shows the time-of-day results: 6 a.m. is the "quietest" time while there are usage spikes at 11 a.m. and 3 p.m.

VI. Conclusion

We have a performed a study on three sizable datasets of Unix command usage. This is the first study to operationalize command expertise and user expertise via several metrics. Based on these metrics, we found several interesting observations on both command and user characteristics that are consistent across datasets, which strengthens our belief that the metrics are stable. Finally, we performed a study on one of the datasets that reveals user behavior and command usage across time of day and day of week.

We believe that our definitions and findings can be used in various scenarios. Our command frequency analysis can be used in real-time to identify outliers, e.g., for masquerade detection. Behavioral patterns can be used to predict Unix user's behavior which helps improve Unix users' experience, from replacing long sequences of commands with scripts to reduce the potential for errors to scaling computing capacity and scheduling support staff. Being able to quantify expertise can be useful in comparing users or assessing how users learn.

Acknowledgments

Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

References

[1] "Usage share of operating systems," https://en.wikipedia.org/wiki/Usage_share_of_operating_systems#Market_share_by_category, Accessed: 2016-03-20.

[2] "Masquerade Attack," https://www.techopedia.com/definition/4020/masquerade-attack, Accessed: 2016-03-20.

[3] M. Schonlau, W. DuMouchel, W.-H. Ju, A. F. Karr, M. Theus, and Y. Vardi, "Computer intrusion: Detecting masquerades," Statistical science, 2001, pp. 58–74.

[4] H.-S. Kim and S.-D. Cha, "Empirical evaluation of svm-based masquerade detection using UNIX commands," Computers & Security, vol. 24, no. 2, 2005, pp. 160 – 168.

[5] R. Maxion and T. Townsend, "Masquerade detection using truncated command lines," in Dependable Systems and Networks, 2002. DSN 2002. Proceedings. International Conference on, 2002, pp. 219–228.

[6] B. Szymanski and Y. Zhang, "Recursive data mining for masquerade detection and author identification," in Information Assurance Workshop, 2004. Proceedings from the Fifth Annual IEEE SMC, June 2004, pp. 424–431.

[7] J. Seo and S. Cha, "Masquerade detection based on svm and sequence-based user commands profile," in Proceedings of the 2Nd ACM Symposium on Information, Computer and Communications Security, ser. ASIACCS '07, 2007, pp. 398–400.

[8] M. Latendresse, "Masquerade detection via customized grammars," in Detection of Intrusions and Malware, and Vulnerability Assessment, ser. Lecture Notes in Computer Science, 2005, vol. 3548, pp. 141–159.

[9] A. Mahajan, "Masquerade detection based on unix commands," master thesis, San Jose State University, 2012.

[10] S. Greenberg, "Using unix: Collected traces of 168 users," Advanced Technologies, The Alberta Research Council, 1988, pp. 1–13.

[11] S. Fitchett and A. Cockburn, "Accessrank: Predicting what users will do next," in Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ser. CHI '12. New York, NY, USA: ACM, 2012, pp. 2239–2242. [Online]. Available: http://doi.acm.org/10.1145/2207676.2208380

[12] R. Chinchani, A. Muthukrishnan, M. Chandrasekaran, and S. Upadhyaya, "Racoon: rapidly generating user command data for anomaly detection from customizable template," in Computer Security Applications Conference, 2004. 20th Annual, Dec 2004, pp. 189–202.