Logging failed authorization attempts and an install request form using the sudo plugin API (milestone)

(title pending)

Tim Ransom

I. Academic Justification

1) Justification for CLI: System administrators have a fondness for command line applications. Takayama et al found that CLI interfaces significantly outrank GUI interfaces in trustworthiness, reliability, robustness, accuracy, and likeability [1]. These results are shown in figure I-1, and show that the general trend of system administrators favoring CLI. This gives us motivation to stick to the traditional text based interface for the development of our sudo plugin. Historical alternatives to Cortisans sudo such as Yahoo's ssu [2] or Tom Christiansen's op [3] also featured command line based interfaces despite the prevelance of GUI's at their respective times of development and adoption.

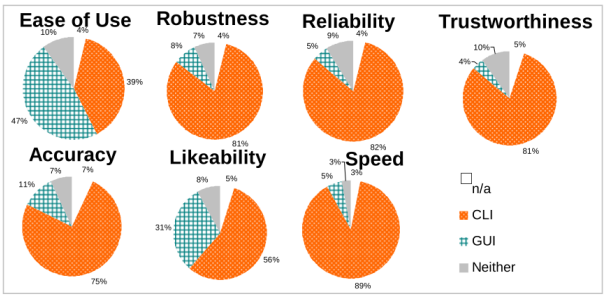

Figure 2. Comparative qualitative judgments of CLIs and GUIs

2) Justification for data collection method: Email is a critical tool for system administrators, for both communication with users and with aggrigating and reporting system information. Look no further than the immense amount of work [4], [5], [6] already present in the security of these messages for their importance to system administrators and the world at large.

II. Progress

I have written the frontend of the code that will communicate with the user. The source code for this can be found at the end of this document in the appendix. This also parses their justification for request to a system command to send the email. This format has been tested on the ada machines part of the SoC public lab. This program will be invoked by the sudo plugin, or integrated into the sudo API.

I have also collected, cleaned, parsed, and analyzed the backlog of fail authorization attempts from my Clemon email account. This generated a database of about 1200 attempts to generate past reports from. Of which over 500 include attempts to use the advanced package tool. Graphs of this data have not been generated but the entries are prepared.

There are a small number of 'moving parts' in this plugin, so the testing primarily consists of witnessing the functionality of the code. Some safety measures are put in place to ensure obvious issues like lack of data do not cause problems; however it should be noted that this application while useful is not necessary for the function of the system. For this reason the testing will be limited to good coding standards rather than full verification and validation required for needed features.

III. Challenges

The sudo API (like many open source projects) lacks clear documentation to specify an entry point for writing a plugin. The script for information gathering is complete but the invocation as a sudo hook has yet to be added. It is likely this will be the most difficult part of this project as the example plugins in the mercurial repository are tightly coupled with the sudo source code.

IV. Bibliography

References

[1] Leila Takayama and Eser Kandogan. Trust as an underlying factor of system administrator interface choice. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1391–1396. Association for Computing Machinery.

[2] Christopher Thorpe. SSU: Extending SSH for Secure Root Administration. page 11.

[3] Tom Christiansen. Op: a flexible tool for restricted superuser access. In *Proceedings of the Workshop on Large Installation Systems Administration III (USENIX Association: Berkeley, CA)*, page 89, 1989.

[4] Scott Ruoti, Jeff Andersen, Scott Heidbrink, Mark O'Neill, Elham Vaziripour, Justin Wu, Daniel Zappala, and Kent Seamons. "We're on the Same Page": A Usability Study of Secure Email Using Pairs of Novice Users. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 4298–4308. Association for Computing Machinery.

[5] Simson L. Garfinkel, David Margrave, Jeffrey I. Schiller, Erik Nordlander, and Robert C. Miller. How to make secure email easier to use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '05*, page 701. ACM Press.

[6] Apu Kapadia. A Case (Study) For Usability in Secure Email Communication. 5(2):80–84.

V. Appendix

A. Frontend source code

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#include <string.h>

int ask_yn_question(char * question) {
```

```
  char answer[8]; // memory is cheap, spring for the extra few bytes for extra cache speed ;)
  printf("%s (yN) ", question);
  scanf("%s", &answer);

  // for ( ; *answer; ++answer) *answer = tolower(*answer);
  for(int i = 0; answer[i]; i++){
    answer[i] = tolower(answer[i]);
  }

  return strcmp(answer, "yes") || strcmp(answer, "y");
}

int get_install_justification(char * justification) {
  printf("Succinctly, why should this package be installed?\n");
  printf("(Hitting enter sends the email, C-c to quit, you have one tweet of space (280 chars)
  scanf("%s", justification);
}

int send_request_email(char * justification) {
  // the user gets 280 characters, the command needs 60
  char mail_command[340] = "mail -s 'request for program install' tsranso@clemson.edu <<< '";

  strcat(mail_command, justification);
  strcat(mail_command, "'");
  printf("%s\n", mail_command);
  printf("email, has been sent :)\n");
  return 0;
}

int main(){
  int response;
  response =
    ask_yn_question("Would you like to submit a request to SoCIT to install this package?");
  if (!response) exit(0);

  response = ask_yn_question("Before we begin, have you considered \
installing this package local to your home directory?");
  if (!response) exit(0);

  char justification[280]; // the length of one tweet
  get_install_justification(justification);

  return send_request_email(justification);
}
```

B. Sending an email with bash

```
mail -s 'test message subject' tsranso@clemson.edu <<< 'message body'
```