CSci487 Penetration Testing Project: AILEE

Grant Haataja
UND Computer Science
Grand Forks, USA
grant.haataja@und.edu

David Wilson
UND Computer Science
Grand Forks, USA
david.andrew.wilson@und.edu

Michael Turnbull
UND Computer Science
Grand Forks, USA
michael.turnbull@und.edu

***Abstract*—This document details the planning, development, and workings of the penetration testing game AILEE, created as a final project for CSCI 487 Penetration Testing class at the University of North Dakota.**

I. Introduction

For this project on penetration testing topics, a hacking simulation game was created. The premise of the game is as follows: the user plays the role of a penetration-testing AI software named AILEE, which stands for Artificial Intelligence Linux Exploit Environment. The game takes place exclusively in a Linux-style terminal environment, with a limited arsenal of commands for the player to use. As the player progresses through the game and "learns" as an AI, the commands available for use increase. Throughout the game, the player is given typed instructions and information from the AI's administrator to assist in learning.

There are two targets to hack in this demo, although there is much potential for expansion. The game uses simulated port scanning, vulnerability scanning, exploitation, and other penetration testing tools to mimic real-life penetration testing methods. Additionally, the game features a storyline with three possible endings, depending on player actions. Special care was taken to handle proper sequence of events.

II. Investigation

A. Planning the Project

Before beginning the development of the game, a suitable platform to run the environment needed to be found. The website Repl.it was decided upon, due to their extensive language support and the ability for multiple people to work simultaneously and have all changes automatically saved to the cloud. [1] The "Multiplayer" mode, as this feature was called, still had a lot of bugs, so forking the project and saving work manually was still necessary, but overall it made the development of AILEE much smoother.

Python3 was selected as the programming language of choice, due to its ease of scripting and strong object-oriented nature. The various classes corresponding to different aspects of the game and environment would be programmed separately, as well as Python scripts for each command available to the player, and every storyline event that could be run. The original plan was for there to be three different targets for the player to hack, but due to time limitations the scope was decreased to two targets.

To enable smooth graphics for the intro screen and the game's ending events, the Python Curses library was referenced and used extensively. [2] This provided the ability to control keyboard input while text displayed on the screen or the ending event graphics played, to increase the smoothness of gameplay.

III. Project Description

A. Intro Screen

For the graphics of the intro screen for the game, ASCII art was used to spell the word AILEE, along with a selection for New Game or Exit. The user can move between the selection using the up or down arrow keys and choose by pressing the enter key. Selecting Exit will cause the terminal session within Repl.it to exit and the game will have to be run again, selecting New Game creates a new session and runs the game.

In addition to these, pressing the up arrow six times in a row will show a hidden third selection, Skip Dialog. This will run the game without displaying any of the instructions and information from the administrator to AILEE, and was very useful for testing the game during development. This mode is not explained or mentioned in the game, as it is not recommended to play without reading the dialogue.

The intro screen makes use of the Python curses library to allow smooth use of the arrow keys keyboard input and prevent buggy graphics.

B. Starting the Game

Upon choosing New Game, the user watches as the administrator logs into their account and launches AILEE.exe to start a new shell. After the shell loads, the first event triggers and text displays on the screen to inform the user what is going on. The administrator gives a brief explanation, and then the user is free to experiment with the Linux-style terminal environment. The terminal runs in the Shell class, (in tandem with the Game and DoStory classes), which supports multiple terminals on various computers. The code for the Shell class is as follows:

```
from termcolor import colored
import replit

import time
import traceback, sys, random

import executables
import events
from MainMenuException import MainMenuException
```

```

DEFAULT_PROMPT = colored("AILEE@{COMP}: {CWD}$ ", 'green')

CMD_NOT_FOUND_STRS = [
  "command not found"
]

class Shell(object):
  """
  Like a seashell.
  """

  def __init__(self, computer, user, agent=None, cwd=None, game=None):
    """
    Create a shell.
    """

    self.computer = computer
    self.user = user
    self.agent = agent
    self.cwd = cwd or computer.fs
    self.prompt = DEFAULT_PROMPT
    self.running = False
    self._command_dictionary = {}
    self.variables = {}
    self.game = game
    self.history = []

    self._setup()

  def _setup(self):
    replit.clear()
    s = "Loading new shell"

    print(s, end='\r')
    i = 0

    # load command dictionary
    for module in executables.__all__:
      self._command_dictionary.update({module: getattr(executables, module).run})
      print(s + '.'*i, end='\r')
      i += 1
      time.sleep(0.1)
    time.sleep(0.3)
    replit.clear()
    #print(constants.title)

  def _get_command_from_str(self, command_str):
    """
    Takes a command name, returns the executable object.
    """

    if command_str == '':
      return False
    if command_str not in self.game.allowed_commands:
      return None
    cmd = self._command_dictionary[command_str]
    return cmd

  def run_command(self, command, args):
    """
    Runs a command.

    Input must be a runnable command that accepts **kwargs.
    """

    command(
      *args,
      computer=self.computer,
      cwd=self.cwd,
      user=self.user,
      agent=self.agent,
      shell=self,
      game=self.game,
    )

  def take_input(self):
    user_input = input(self.prompt.format(
      COMP=str(self.computer.name),
      CWD=str(self.cwd),
      USER=self.user),
    )

    parts = [p.strip() for p in user_input.split(' ')]
    command = parts[0]
    args = parts[1:]

    return command, args

  def one_command(self):
    command, args = self.take_input()
    cmd = self._get_command_from_str(command)
    if cmd is None:
      self.cmd_not_found()
      return
    elif cmd is False:
      return  # nothing on empty commands
    cname = cmd.__module__.split('.')[-1]
    if not cname == 'doStory':
      self.game.history.append([cname, args])
      self.history.append([cname, args])

    if not (command or args):
      return  # skip empty input
    self.run_command(cmd, args)

  def halt(self):
    self.running = False

  def cmd_not_found(self):
    self.history.append([None, []])
    self.game.history.append([None, []])
    print("Command not found")
    #print(random.choice(CMD_NOT_FOUND_STRS))

  def start_shell_loop(self):
    self.running = True
    while self.running:

      # This is the line of code that integrates the story VVV


      try:
        self.run_command(events.doStory.run, [])
        self.one_command()
      except KeyboardInterrupt:
        print()
        #print("\nYou can't leave! ", end='')
      except KeyError as e:
        self.cmd_not_found()
      except AssertionError as e:
        print(str(e))
      except MainMenuException:
        raise MainMenuException
      except Exception as e:
        print(colored("Something went wrong.  I'm not quite sure what.  Maybe try again?", 'red'))
        # Uncomment VV for full tracebacks
        #einfo = sys.exc_info()
        #traceback.print_exception(*einfo)
```

The user is encouraged to try out the various possible commands, which can be displayed using the *help* command. The story continues after the user has ran ten commands, (they can be the same or different commands, it doesn't matter).

C. Equations

Number equations consecutively. To make your equations more compact, you may use the solidus (/), the exp function, or appropriate exponents. Italicize Roman symbols for quantities and variables, but not Greek symbols. Use a long dash rather than a hyphen for a minus sign. Punctuate equations with commas or periods when they are part of a sentence, as in:

$$a + b = \gamma \tag{1}$$

Be sure that the symbols in your equation have been defined before or immediately following the equation. Use "(1)", not "Eq. (1)" or "equation (1)", except at the beginning of a sentence: "Equation (1) is . . ."

D. LaTeX-Specific Advice

Please use "soft" (e.g., `\eqref{Eq}`) cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please don't use the `{eqnarray}` equation environment. Use `{align}` or `{IEEEeqnarray}` instead. The `{eqnarray}` environment leaves unsightly spaces around relation symbols.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BibTeX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BibTeX to produce a bibliography you must send the .bib files.

LaTeX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LaTeX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Do not use `\nonumber` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

E. Some Common Mistakes

- The word "data" is plural, not singular.
- The subscript for the permeability of vacuum μ_0, and other common scientific constants, is zero with subscript formatting, not a lowercase letter "o".
- In American English, commas, semicolons, periods, question and exclamation marks are located within quotation marks only when a complete thought or name is cited, such as a title or full quotation. When quotation marks are used, instead of a bold or italic typeface, to highlight a word or phrase, punctuation should appear outside of the quotation marks. A parenthetical phrase or statement at the end of a sentence is punctuated outside of the closing parenthesis (like this). (A parenthetical sentence is punctuated within the parentheses.)
- A graph within a graph is an "inset", not an "insert". The word alternatively is preferred to the word "alternately" (unless you really mean something that alternates).
- Do not use the word "essentially" to mean "approximately" or "effectively".
- In your paper title, if the words "that uses" can accurately replace the word "using", capitalize the "u"; if not, keep using lower-cased.
- Be aware of the different meanings of the homophones "affect" and "effect", "complement" and "compliment", "discreet" and "discrete", "principal" and "principle".
- Do not confuse "imply" and "infer".
- The prefix "non" is not a word; it should be joined to the word it modifies, usually without a hyphen.
- There is no period after the "et" in the Latin abbreviation "et al.".
- The abbreviation "i.e." means "that is", and the abbreviation "e.g." means "for example".

An excellent style manual for science writers is [**?**].

F. Authors and Affiliations

The class file is designed for, but not limited to, six authors. A minimum of one author is required for all conference articles. Author names should be listed starting from left to right and then moving down to the next line. This is the author sequence that will be used in future citations and by indexing services. Names should not be listed in columns nor group by affiliation. Please keep your affiliations as succinct as possible (for example, do not differentiate among departments of the same organization).

G. Identify the Headings

Headings, or heads, are organizational devices that guide the reader through your paper. There are two types: component heads and text heads.

Component heads identify the different components of your paper and are not topically subordinate to each other. Examples include Acknowledgments and References and, for these, the correct style to use is "Heading 5". Use "figure caption" for your Figure captions, and "table head" for your table title. Run-in heads, such as "Abstract", will require you to apply a style (in this case, italic) in addition to the style

provided by the drop down menu to diff
the text.

Text heads organize the topics on a
basis. For example, the paper title is
because all subsequent material relates
one topic. If there are two or more
level head (uppercase Roman numeral
conversely, if there are not at least tv
subheads should be introduced.

H. Figures and Tables

a) Positioning Figures and Tabl
tables at the top and bottom of colum
in the middle of columns. Large figure
across both columns. Figure captions
figures; table heads should appear at
figures and tables after they are cited
abbreviation "Fig. 1", even at the begi

TABLE I
TABLE TYPE STYLE

Table Head	Table Column H	
	Table column subhead	*Su*
copy	More table copy[a]	

[a]Sample of a Table footnote.

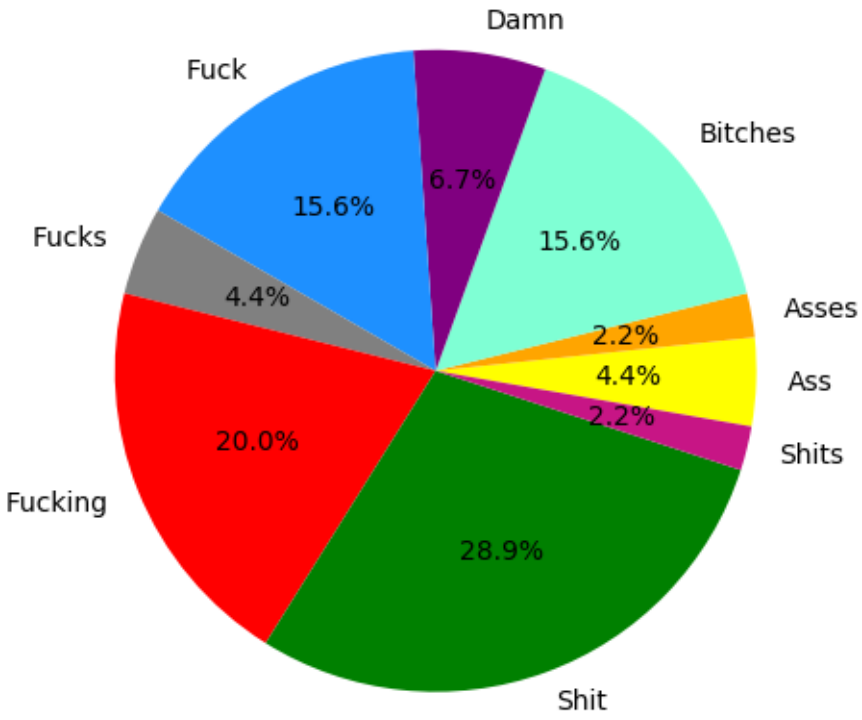

Fig. 1. Example of a figure caption.

Figure Labels: Use 8 point Times N
labels. Use words rather than symbols
writing Figure axis labels to avoid conf
example, write the quantity "Magnetization", or "Magnetization, M", not just "M". If including units in the label, present them within parentheses. Do not label axes only with units. In the example, write "Magnetization (A/m)" or "Magnetization {A[m(1)]}", not just "A/m". Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)", not "Temperature/K".

ACKNOWLEDGMENT

The preferred spelling of the word "acknowledgment" in America is without an "e" after the "g". Avoid the stilted expression "one of us (R. B. G.) thanks ...". Instead, try "R. B. G. thanks...". Put sponsor acknowledgments in the unnumbered footnote on the first page.

REFERENCES

Please number citations consecutively within brackets [1]. The sentence punctuation follows the bracket [2]. Refer simply to the reference number, as in [**?**]—do not use "Ref. [**?**]" or "reference [**?**]" except at the beginning of a sentence: "Reference [**?**] was the first ..."

Number footnotes separately in superscripts. Place the actual footnote at the bottom of the column in which it was cited. Do not put footnotes in the abstract or reference list. Use letters for table footnotes.

Unless there are six authors or more give all authors' names; do not use "et al.". Papers that have not been published, even if they have been submitted for publication, should be cited as "unpublished" [**?**]. Papers that have been accepted for publication should be cited as "in press" [**?**]. Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [**?**].

REFERENCES

[1] Repl.it, "The world's leading online coding platform," repl.it. [Online]. Available: https://repl.it/site/features. [Accessed: 02-May-2019].
[2] A. M. Kuchling and E. S. Raymond, "Curses Programming with Python¶," Curses Programming with Python - Python 3.7.3 documentation. [Online]. Available: https://docs.python.org/3/howto/curses.html. [Accessed: 02-May-2019].