



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

2022-03

**DEVELOPING AN EXPANDABLE GUI TOOL TO
ENHANCE NETWORKING EDUCATION:
GRAPHICAL USER INTERFACE FOR SHELL
ENTRY (GUISE)**

Anderson, Clinton J.

Monterey, CA; Naval Postgraduate School

<https://hdl.handle.net/10945/69609>

Copyright is reserved by the copyright owner.

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>



**NAVAL
POSTGRADUATE
SCHOOL**

MONTEREY, CALIFORNIA

THESIS

**DEVELOPING AN EXPANDABLE GUI TOOL TO
ENHANCE NETWORKING EDUCATION: GRAPHICAL
USER INTERFACE FOR SHELL ENTRY (GUISE)**

by

Clinton J. Anderson

March 2022

Thesis Advisor:
Co-Advisor:

Amela Sadagic
John D. Fulp

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.			
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March 2022	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE DEVELOPING AN EXPANDABLE GUI TOOL TO ENHANCE NETWORKING EDUCATION: GRAPHICAL USER INTERFACE FOR SHELL ENTRY (GUISE)			5. FUNDING NUMBERS
6. AUTHOR(S) Clinton J. Anderson			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A
13. ABSTRACT (maximum 200 words) The lack of systemic education dedicated to computer networks and the general inadequacy of students' comprehension of the structure and the dynamics of the networks are arguably issues in most public schools. In the situation where the internet is a commodity, the increase in the threat of global attacks on many computing resources is exceptionally high, and so is the consequential importance of the global cyber workforce. Most people achieve their basic understanding through their routine use of computers at home, and it is both pragmatic and more effective to consider using basic home tools because of their didactic benefits. We designed and developed GUISE (GUI for Shell Entry) as an intuitive interface that makes command entry and network analysis easier for masses of users. GUISE leverages the operating system's capabilities and allows inexperienced users with no expertise in the computer networking domain to acquire enhanced network situational awareness in a competent manner. The ultimate benefit of the GUISE tool is providing its users with an educational aspect focused on the networking elements of their home computer infrastructure. That approach has the potential to directly support the growth of their networking literacy and proficiency, and their navigation of the networking landscape with enhanced confidence and safety.			
14. SUBJECT TERMS network analysis, visualization, user interface			15. NUMBER OF PAGES 99
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**DEVELOPING AN EXPANDABLE GUI TOOL TO ENHANCE NETWORKING
EDUCATION: GRAPHICAL USER INTERFACE FOR SHELL ENTRY (GUISE)**

Clinton J. Anderson
Civilian, Scholarship for Service
BA, University of California, Berkeley, 2013
MS, University of Southern California, 2018

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
March 2022**

Approved by: Amela Sadagic
Advisor

John D. Fulp
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The lack of systemic education dedicated to computer networks and the general inadequacy of students' comprehension of the structure and the dynamics of the networks are arguably issues in most public schools. In the situation where the internet is a commodity, the increase in the threat of global attacks on many computing resources is exceptionally high, and so is the consequential importance of the global cyber workforce. Most people achieve their basic understanding through their routine use of computers at home, and it is both pragmatic and more effective to consider using basic home tools because of their didactic benefits. We designed and developed GUISE (GUI for Shell Entry) as an intuitive interface that makes command entry and network analysis easier for masses of users. GUISE leverages the operating system's capabilities and allows inexperienced users with no expertise in the computer networking domain to acquire enhanced network situational awareness in a competent manner. The ultimate benefit of the GUISE tool is providing its users with an educational aspect focused on the networking elements of their home computer infrastructure. That approach has the potential to directly support the growth of their networking literacy and proficiency, and their navigation of the networking landscape with enhanced confidence and safety.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	PROBLEM SPACE	1
B.	RESEARCH QUESTIONS.....	2
C.	MOTIVATION	3
	1. Home or Job Situational Awareness	3
	2. Educational Maturity	4
	3. Indication of Compromised Systems.....	4
D.	SCOPE	6
E.	METHODOLOGY	6
F.	THESIS STRUCTURE	7
G.	CHAPTER SUMMARY.....	8
II.	BACKGROUND AND LITERATURE REVIEW	9
A.	INTRODUCTION.....	9
B.	RELATED CYBER INCIDENTS.....	9
	1. DarkSide Ransomware Attack on Colonial Pipeline.....	9
	2. Hackers Poison Water Plant Remotely.....	10
C.	CYBER PEDAGOGY THROUGH APPS.....	10
	1. Office Minefield.....	11
	2. Packet Tracer Simulation Tool as Pedagogy to Enhance Learning of Computer Science Concepts	12
D.	COMMAND LINE INTERFACE (CLI) / POWERSHELL.....	13
E.	CHAPTER SUMMARY.....	15
III.	NETWORK MONITORING TOOLS.....	17
A.	INTRODUCTION.....	17
B.	TOOLS FOR HOME USE.....	18
	1. Microsoft Network Status	18
	2. Apple AirPort Utility	19
	3. Cross-comparison	20
C.	COMMERCIAL NETWORK MONITORING TOOLS.....	21
	1. Wireshark (with MaxMind Geolocation)	21
	2. TShark	23
	3. Network Performance Monitors: SolarWinds, Dynatrace, PRTG Analyzer	24
	4. Cross-comparison	26
D.	TOOLS FOR EDUCATION.....	26

1.	Books Dedicated to Networking	26
2.	University Classes	27
3.	Internet Search and Tutorials.....	27
4.	Cisco Packet Tracer	27
E.	CHAPTER SUMMARY.....	28
IV.	USER INTERFACE DESIGN.....	31
A.	INTRODUCTION.....	31
B.	PROTOTYPE IDEATION: PERSONAL CYBER- CONNECTIVITY PICTURE (PCCP).....	33
1.	PCCP Assorted Applications	34
2.	PCCP Initial Command Prompt Display	35
3.	PCCP Transition to GUISE	37
C.	DESIGN OF GUISE DISPLAY.....	38
D.	GUISE LAYOUT	41
E.	GUISE MENUS AND USER INTERACTIONS	43
1.	Commands Menu	43
2.	Services Menu.....	45
3.	Simple and Parameterized Options Menus	45
4.	Manual Text Entry	46
5.	Tooltips.....	47
F.	VISUALIZATION OF GUI ELEMENTS.....	47
1.	Unity as a Development Environment	47
2.	GUISE Features	47
3.	Options Layout.....	48
4.	Color Scheme and Sizing.....	48
5.	GUISE GUI Layout and Operation	50
G.	CHAPTER SUMMARY.....	53
V.	SYSTEMS INTEGRATION AND IMPLEMENTATION.....	55
A.	INTRODUCTION.....	55
B.	GUISE C# DEVELOPMENT IN UNITY.....	55
C.	NAMING CONVENTIONS.....	56
D.	JSON DATA	57
E.	DATALOADER AND DATAVISUALIZER SCRIPTS	58
F.	STATUSNODES: CONTEXT FOR INDEXES, PAGES, AND GROUPS.....	60
1.	Indexes	61
2.	Pages.....	61
3.	Groups.....	62

G.	BACKSPACE AND CONTEXT SWITCHING	63
H.	TOOLTIPS AND TOOLTIP TRIGGERS	63
I.	OUTPUT TO FILE / INPUT TO SCREEN	64
J.	CHAPTER SUMMARY.....	65
VI.	CONCLUSIONS AND RECOMMENDATIONS.....	67
A.	PRINCIPAL THESIS CONCLUSIONS	67
B.	ADDRESSING RESEARCH QUESTIONS.....	67
	1. Research Question 1: Interface for enhanced situational awareness	67
	2. Research Question 2: Application that is extendible.....	68
C.	LIMITATIONS	70
D.	RECOMMENDATIONS FOR FUTURE USE	71
	1. GUISE as a Security Management Tool.....	71
	2. GUISE as an Educational Tool.....	72
	3. Microsoft Windows Integration.....	72
	4. Open-Source Project.....	73
E.	FUTURE WORK.....	73
	1. Usability Study	73
	2. User Study Focused on Learning Effectiveness	73
	3. Implementation on Other Operating Systems	73
	LIST OF REFERENCES	75
	INITIAL DISTRIBUTION LIST	79

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1.	Malwarebytes home. Source: BECS Software Solutions (2020).....	5
Figure 2.	Malwarebytes' after-action report. Source: BECS Software Solutions (2020).....	5
Figure 3.	Changing directories in Command Prompt and PowerShell. Source: Hoffman (2017).	14
Figure 4.	Nmap, a networking command uncommon to normal users. Source: Wouk (2021).	15
Figure 5.	Microsoft Network Status.....	19
Figure 6.	Apple AirPort Utility's cursory yet "pretty" interface. Source: Lee (2016).....	20
Figure 7.	Wireshark showing extensive contents of a full packet. Source: Wireshark (n.d.).	22
Figure 8.	TShark showing computer interfaces. Source: Kothari (2021).	23
Figure 9.	SolarWinds screen showing various networking elements. Source: SolarWinds (2022).....	24
Figure 10.	Dynatrace with depictions of cloud and network services. Source: Dynatrace (2022).	25
Figure 11.	PRTG Analyzer showing worldwide network status. Source: Paessler (2022).....	25
Figure 12.	Packet Tracer featuring 4 LANs and their connections. Source: Cruz Sendy (2015).....	28
Figure 13.	Idea for MS Network Status opening applications graphically	34
Figure 14.	Four Main Requirements of PCCP	35
Figure 15.	PCCP options, tooltips, flow, and end	36
Figure 16.	Jet fighter cockpit interior Source: World War Wings (2022).	39
Figure 17.	Flight simulator in-game screen vs. initial GUISE layout Source: Kellerer et al. (2011).....	40

Figure 18.	Early PCCP Mockup #1: Showing depressed (grey) buttons, and how simple / parameter options and tooltips work	41
Figure 19.	PCCP Mockup #2: Showing tooltips (in purple) on options, replacing separate “usage” button.....	42
Figure 20.	PCCP Mockup #3: Showing InputField to replace parameter buttons, and TargetName to input strings.....	42
Figure 21.	Later PCCP version with newer tooltips, arrow bars, TargetName, and InputField	43
Figure 22.	GUISE blueprint	43
Figure 23.	Ping using /? to show its help menu, options, and their descriptions	45
Figure 24.	Color scheme of GUISE buttons.....	49
Figure 25.	Button positioning based on Command 0	50
Figure 26.	Final GUISE interface.....	51
Figure 27.	GUISE after running a command, hovering over an option	52
Figure 28.	GUISE UI elements within the Unity game engine.....	56
Figure 29.	Expanded menu (left) and collapsed method conventions (right)	57
Figure 30.	Overview of JSON Fields	58
Figure 31.	Expanded JSON data showing fields for commands and options	58
Figure 32.	StatusNode Layout showing Groups, Pages within Groups, and Indexes within Pages.....	61
Figure 33.	Dynamic Tooltips extending downwards (left) and to center (right)	64

LIST OF TABLES

Table 1.	GUISE sizes and colors	49
Table 2.	Datavisualizer methods and their descriptions	59

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

CLI	command line interface
GUI	graphical user interface
GUISE	Graphical User Interface for Shell Entry
HTML	hypertext markup language
IP	internet protocol
LAN	local area network
MAC OS X	Apple Operating System
MS	Microsoft
OS	operating system
OSI	Open Sources Interconnection
PCCP	Personal Cyber-Connectivity Picture
UI	user interface
URL	Uniform Resource Locator

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

This material is based upon activities supported by the National Science Foundation under Agreement No 1565443. Any opinions, findings, and conclusions or recommendations expressed are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

I would like to thank Dr. Sadagic for taking me on as a student, giving me access to the lab room (tons of long nights spent there), and sharing so much of your knowledge to make me a better UI developer and more of a professional. Your bright spirits and constant suggestions were inspiring. I also give my thanks to J.D. Fulp, my co-advisor, the Naval Postgraduate School's best teacher and just an awesome person to be around. Both of you were outstanding advisors, mentors, and co-authors on this thesis—the most intensive and comprehensive project I have ever worked on in my life. With your specific direction and exceptional guidance, you made this project challenging while at the same time fun. It was a blast to work with you two. I wish you the best with your future students, with your remaining time at NPS, and in your personal lives.

Eric, you have been outstandingly supportive, as you always have been during my entire academic career, as were Haipeng and Haiyan, my Monterey host family. My SFS cohort, Andrea, Sutter, and Elyssa, and the other members of the Fantastic Four, Aria and Josh—thank you for sharing this ride and for being so encouraging. I also want to thank the SFS faculty at NPS, including Dr. Irvine, Dr. Davis, and Cecelia Davis for showing nothing but support from day one of arriving in Monterey as well as Michele and other TPO staff. Though most of it was online due to the Covid-19 pandemic, you all showed high professionalism getting all the students through during that unconventional time.

Finally, I would like to thank the National Science Foundation and the other staff and teachers at the Naval Postgraduate School for making the Monarch Scholarship possible. I came to NPS to learn, grow, and make myself a better programmer and cyber specialist. I am leaving NPS even more confident than I had hoped to.

Go NAVY! Go BEARS! Fight On!

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. PROBLEM SPACE

Analyzing network connections and network status is a daunting task for even experienced network analysts and non-specialists. The average home personal computer user or novice network student has little to no knowledge of the jargon, application, or operation of the fundamental protocols or tools used within the networking domain. Common questions such as, “How does a router work?,” “What is an IP Address?,” and “How many computers are connected to my LAN?” are answered by an average user superficially, at best.

Just as public knowledge about cars and engines has grown with the progression of the automobile industry, so too should fundamental networking knowledge grow among public computer users. Why? Benefits range from helping a home user understand what malware is and how to prevent it, increasing the baseline knowledge of job seekers interested in joining the cyber workforce, and providing more effective support and initial training to cyber students. Progress toward this goal could begin by instituting a solid picture of the fundamental networking concepts—What is a router? What are the network stacks? How do machines communicate through a network? How does the Domain Name System work? Cybersecurity attacks within American companies are at an all-time high and are set to increase, becoming one of the most exposed and prolific methods of attack on the country. Thus, inculcating greater understanding among the growing number of network users should help mitigate this risk.

Many of the topics from the cybersecurity realm are explained in the literature to computer users, yet many have no interest in learning about the concepts, likely due to two main reasons. The first is that the introduction of some of the interfaces is presented in a complicated way, scaring newcomers off. Secondly, few of the underlying technical concepts are presented to average users via their daily use of computing platforms to develop their growth. Though some of the underlying concepts can be complicated to understand, we feel that fundamental improvement in situational awareness can be

achieved if the neophyte home user is provided with an easy-to-use way to interact with their computer's ongoing state of connectivity.

This thesis examines the design and use of a home-use shell input method, and by extension, a network analysis tool when scripted with the right commands, that leverages the operating system's capabilities and allows naïve users who have no expertise in the computer networking domain to use advanced capabilities in a more intuitive and easier-to-use format. The feasibility study at the center of this research effort seeks to design and develop an in-house network analysis tool, provisionally named GUISE (GUI for Shell Entry). This tool's capability is achieved by wrapping a range of OS commands in a graphical user interface (GUI). The tool allows the users to select different commands in a full range of their capabilities and, over time, learn more about them. The in-house, expandable tool includes a simple-to-use GUI with embedded help features and delivers enhanced network situational awareness to the user.

The primary purpose of this effort is to improve the efficiency of the network monitoring tasks that learners and forensic analysts face and improve the overall understanding of the relations and causalities of events related to network operations. Our attention is on naïve users—people unfamiliar with the operating system (OS) commands, command line, or commercial tools used to analyze network status. The same tool can be used by users who understand OS commands but need to refresh that knowledge without using manuals.

B. RESEARCH QUESTIONS

The research questions addressed in this thesis are:

- What type of graphical user interface and user interactions can provide enhanced comprehension of OS commands and their capabilities to novice users?
- Could the underlying system architecture and user interface be designed to support a variety of classes of OS commands and services beyond the network analysis?

C. MOTIVATION

Network capability is manifested in different forms depending on the type of user. In general, most people can benefit from short and non-complicated network tasks, and the tool designed for this thesis can apply to all these users. More professional users need a distinct, comprehensive set of tools, which requires that our tool be extendable to support their needs while not confusing the less technically savvy users.

1. Home or Job Situational Awareness

Most home users have limited knowledge of what happens behind the scenes of their system and internet browser. When asked what happens when one connects to the internet, in general, there may be some knowledge of interacting with a URL or search engine, and some even have knowledge of web HTML when it is presented. Virus scanners tend to be known to be operating silently in the background. The plane of these few facts represents a good portion of the average user's knowledge base.

More advanced users use precautions such as a "firewall" to block undesired incoming traffic, use a "proxy" for anonymous searches, use secure settings for outbound searches, and apply settings to prevent "cookies." These terms (firewall, proxy, cookies, etc.) represent substantial professional ability. However, general usage and the modification of these settings are not common knowledge.

Many workplaces do not have enterprise-level cybersecurity systems that can detect, modify, and change network-connections in an easy-to-understand fashion. While big companies have an expert who defends their computer systems, smaller companies or those without a dedicated security employee do not, and they stand to benefit from having access to tools that make their jobs easier.

A tool like the one in this thesis can provide unskilled workers with some selected expert abilities, can introduce services that were unknown before, and serve to gradually escalate knowledge to increase home cyber defense or present an additional layer of defense for the workplace.

2. Educational Maturity

While our tool has a general value and it can be useful to anyone who picks it up, it is best suited for those who have a necessity for being involved in networking, particularly:

- Naïve Users—Home users and average computer users at most jobs are the most populous while also being the ones who need the most assistance.
- 1st-Year College Students—Particularly those studying cybersecurity or computer science.
- Cyber Professionals—This tool can mainly be used for those who could use the quick GUI for simple or trivial tasks. These professionals can also write scripts for continuous defense tasks and provide those scripts to non-technical company employees.

The United States’ adversaries in this field tend to have training programs, with numbers seeming to increase dramatically as years continue.

3. Indication of Compromised Systems

Procmon, Regmon, and Sysinternals are also comprehensive networking and processing tools that are widely regarded by our limited cybersecurity engineer force (Russinovich, 2022). These applications provide more than enough capability to perform almost any task, such as malware tracking and network monitoring.

When something does go wrong, users need to be aware of those events. Problem examples include compromised systems (whether malicious or not), unauthorized accesses, or uploading of malware. A virus detection and removal program like Malwarebytes can help (Malwarebytes, 2022). It works out of the box by simply downloading the software and clicking a single “Scan” button, with images shown in Figure 1 and Figure 2:

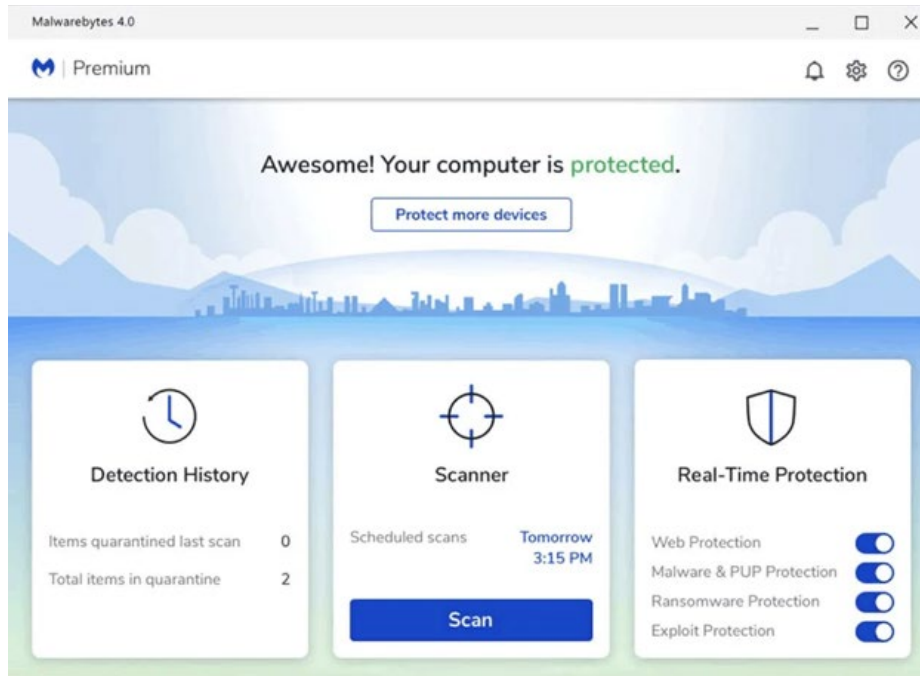


Figure 1. Malwarebytes home.
Source: BECS Software Solutions (2020).

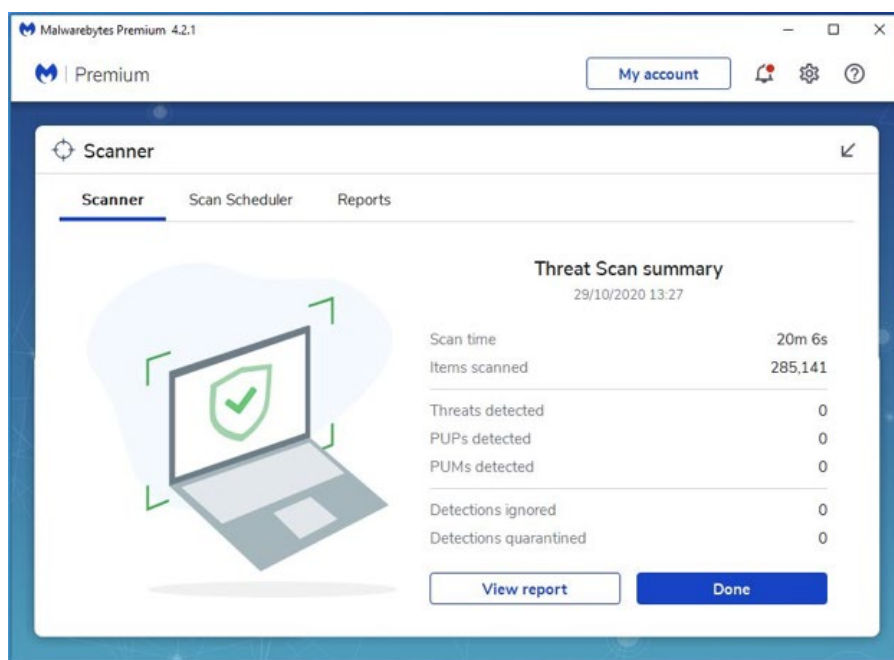


Figure 2. Malwarebytes' after-action report.
Source: BECS Software Solutions (2020).

Initially, the users can run Malwarebytes as a first step toward networking knowledge progression. Though simplistic, even the use of this simple one-button program introduces keywords such as scanning, ransomware, PUPs (potentially unwanted programs) and PUMs (potentially unwanted modifications). They may notice differences in traffic, increased CPU usage, or recognize an unknown application in the Task Manager. Using this program, do users have the tools to really uncover the root of a problem? Usually not.

Professional tools like Sysinternals are designed for professionals, and although virus scanners are a great first step as they are better than nothing, they tend to be too simplistic in their delivery and a deeper dive is needed. Our application can slowly introduce how to find a virus by using internal tools native to the operating system while also providing a host of options that are not usually provided with superficial scanner software like Malwarebytes.

D. SCOPE

The scope of this feasibility study is to design and prototype a home-use network analysis tool that allows naïve users to employ a range of OS commands and acquire an enhanced network situational awareness.

E. METHODOLOGY

In recent years networking technologies have advanced rapidly, yet the common person's understanding of how a network works has not grown with this trend. This thesis will systematically review the frameworks that knowledge can be presented in within common operating systems, and then design and develop an application that can be integrated fluidly. Specifically, we will execute the following tasks:

1. Conduct literature review.
2. Execute an analysis and study of networking tasks.
3. Create a conceptual design of the system and identify the capabilities that will be supported.

4. Identify all system requirements for developing and prototyping, including research on prototype design and Application Programming Interfaces (APIs).
5. Design and implement the prototype tool and associated GUI.
6. Identify lines of future work and produce recommendations for potential approaches.

F. THESIS STRUCTURE

Chapter II provides several highlights of the recent cyber-attacks against the U.S. and addresses the necessity of an easy-to-use network monitoring tool. It compares and discusses existing technology and highlights their advantages and disadvantages regarding the thesis purposes.

Chapter III provides an overview of the current monitoring tools found on the major operating systems, Windows Home and Mac OS X, and the two main ways to monitor traffic, overviewing the native OS command line and current commercial software.

Chapter IV outlines the design goals to achieve the desired results of this effort and presents the elements of GUISE tool design—the layout of its elements and user interactions.

Chapter V elaborates the elements of the system development, including listing specifics within the JSON modules, showing how they relate to script classes, describing the naming of elements, depicting flowcharts of methods and variables, and the interconnection with the Unity interface.

Chapter VI highlights the conclusions of the work on this thesis and suggests future work.

G. CHAPTER SUMMARY

Chapter I covered the problem space addressed by this thesis, the specific research problem that was examined, motivation for the work, research questions, scope of the thesis, and the methodology that was used to address the research questions.

II. BACKGROUND AND LITERATURE REVIEW

Cyber-attacks have become rampant in recent years due to the ease with which they can be accomplished, the anonymity of the attackers, and the relative gains that can be procured.

A. INTRODUCTION

Recent 2020–2021 cyber incidents have taken the U.S. by surprise. The attacks have shown how vulnerable the U.S. economy is to simple hacking methods that cost a fraction of the damage done. The following examples illustrate how anonymous users can cause major damage with very little attribution.

B. RELATED CYBER INCIDENTS

1. DarkSide Ransomware Attack on Colonial Pipeline

DarkSide is a hacking group that caused a major upset in the U.S. in May of 2021 when it caused a ransomware attack on Colonial Pipeline. Colonial Pipeline owns a 5,500-mile East Coast oil pipeline that controls up to 45% of the fuel on the coast, and because DarkSide had exfiltrated shared internal data, the decision was made to shut down the system until resolution (Bing, 2021). The prices of oil spiked until a 5 million dollar settlement was made (Wilke, 2021).

Despite the sophistication of the attack, the root cause was a case of a normal employee who was unfamiliar with the benefits of up-to-date patching, safely copying, and performing correct backups. Additionally, the case may be more elementary than that. According to Reuters, an employee password was stolen, and the company did not use multi-factor authentication (Kelly & Resnick-ault, 2021).

Training for these situations is a staple of any cyber defense coordinator, and the fact that so much damage could have been caused by such a simple mistake is indefensible. This problem has an easily correctable fix that we must ensure becomes implemented among the public.

2. Hackers Poison Water Plant Remotely

According to CNN, an unknown culprit gained entry to the IT systems at a water treatment facility in the small town of Oldsmar, Florida. The hackers were able to take control of the water system utilities with “TeamViewer,” a remote access tool that had not seen usage by the plant in over six months. Using remote software controls, they increased the sodium hydroxide chemicals in the water to dangerous levels, up to 100 times normal (Marquardt et al., 2021).

The attack was caught in time, and no damage was done. However, an attack like this highlights the need for increased user awareness of common cyber-attacks, their methods from simple to complex, the threat opportunities that emerge from remote access, and the absolute necessity to monitor a network by a non-expert.

C. CYBER PEDAGOGY THROUGH APPS

Starting from a very young age, humans experience many activities that are complicated to grasp, yet due to continual and pervasive operation, the functions are learned without ever going through formal instruction. Some of these activities are swimming, using Microsoft Excel, or even reading a roadmap. Once learned, the actions seem simple enough to be trivial, but for a person new to the activity—say a landlocked individual for swimming or an older person who is new to computers in general—their first interaction with the activity might be enough of an obstacle to make them quit altogether or make them consider it too difficult to learn enough to be competent.

When considering technical activities, like programming mathematical functions in Microsoft Excel spreadsheets, what makes systems so easy to learn is the design that goes into providing the system through “affordance” (Gibson & Carmichael, 1966). When some system should be used a certain way (as in you expect certain system behaviors), the system is designed such that the interaction seems natural. A key portion of this is the word ‘natural,’ meaning no instruction manual or assistance outside of the application is necessary. The effort to accomplish the goal is not hindered by the complication of its usage.

1. Office Minefield

A few applications come to mind when considering ease of use coming inherently from good design. In Joel Garreau's article "Office Minefield," he discusses two games that come pre-installed on Microsoft, Solitaire and Minesweeper, to surreptitiously teach new computer users the basic function of navigating a computer, such as how to use a mouse with its left and right clicks, how to navigate to small icons, and how to use the various menus (Garreau, 1994). Even the fact that you had a timer, which means you had to do these actions quickly, added to the expertise being built with each play. Lizzy Duzan, a lead product manager for entertainment, stated that, "it soothed people intimidated by the operating system, ... giving them something fun to do while also teaching them how to use a mouse" (Garreau, 1994).

If one was to ask a common computer user how or even when they learned how to use the computer, including how to use a mouse, type in Excel, or navigate the operating system, many may not be able to recall it. That demonstrates a good design.

Unfortunately, the same situation could be said of command prompt usage. Although being capable of swiftly moving around in that environment would be highly beneficial for most if not all computer users, the 'intimidation of the system' shuns many non-technical people away. Although hard statistics could not be found on how few people use the command line, it is not out of the question to think that fewer people use it on a regular basis than those who do.

One reason for this might be that outside of performing computer security, there just are not that many requirements to use the terminal. Many applications can be downloaded online and immediately installed through GUI windows. Even still, many network features such as connecting to the internet or virus scanning are simply performed by the operating system with almost no interaction from the user, except for a few button clicks to acknowledge warnings or perform updates.

What is lost in that ease of access is the user's systemic capability to really use their computer, which is learned through consistent use the shell, i.e., the command line. We therefore exchange expertise of our system—forgoing memorizing folder locations,

account hierarchies, and commands and utilities—to save time and confusion. This lack of knowledge eventually hurts normal workers and home users later in life when they are faced with a scarce but destructive cyber-attack.

2. Packet Tracer Simulation Tool as Pedagogy to Enhance Learning of Computer Science Concepts

Teacher and researcher Dr. Vijayalakshmi begins his topic introduction by stating, “...a working knowledge of networking is essential to computer science graduates—an opinion put forth in the Computer Engineering Curricula 2016-ACM guidelines” (Vijayalakshmi et al., 2016). This baseline education in networking is therefore (or rather, should be) required for anyone going into cyber, such as the security analysts, IT techs, and researchers so far discussed. He implies that students would do well if their short time spent studying in upper-level security courses were spent on advanced techniques and practice, rather than spending it on fundamentals. He also states that the imagination level of students should be high as well. That is important because networks cover vast bodies of knowledge and experience, and to keep track of it all requires a healthy dose of system knowledge, practical experience, and mental visualization.

Any course covering networking in sufficient detail requires a fundamental understanding of diverse subjects such as electronic devices, software concepts and communication technology. Dr. Vijayalakshmi used a tool called Cisco Packet Tracer for his research, which covered that knowledge base well. After guiding the students in his university class through using the tool on fundamental topics like demonstrating network commands, interpreting ping and traceroute output, troubleshooting and resolving network issues, examining the functionality of TCP, UDP, and application layer protocols, Dr. Vijayalakshmi found that students had an 85% attainment of objectives, a significant increase from around 65% before the activity.

The idea behind this is that even though Packet Tracer is a GUI, and the provided exercises are geared to classroom education, once students are familiar with a basic network setup and can visualize different configurations in their head, swapping to a text-based approach will not be as daunting.

D. COMMAND LINE INTERFACE (CLI) / POWERSHELL

According to Lifewire.com, the Command Prompt is a particularly useful program to users and administrators to execute commands. It can be opened through the Start Menu, by using the Run command, or by opening it from its folder location, usually C:\Windows\system32\cmd.exe. Officially called “Windows Command Processor,” it is also known as the command shell or simply the command prompt. “It’s used to execute entered commands. Most of those commands automate tasks via scripts and batch files, perform advanced administrative functions, and troubleshoot or solve certain kinds of Windows issues” (Fisher, 2022). Using a CLI is preferred over a GUI among developers and administrators, particularly when developing software, configuring hardware, or using the extended capabilities that the GUI developers had not included in their version (Campion, 2020).

Windows has two command shells: the basic familiar Command shell that has existed for decades (with a black background), and an upgraded version with advanced scripting language capabilities and additional features (with a blue background), called PowerShell (Microsoft | Docs, 2022).

According to How-to-Geek’s Chris Hoffman, despite PowerShell being more complicated, requiring more memorization and incorporating new elements like ‘cmdlets’ to be learned, it is preferred over the traditional Command shell because it allows access to a host of expanded features, and users can be more explicit. Windows began including PowerShell on all Windows operating systems since Windows 7 and has since become the default choice for inputting commands through a shell since Windows 10 (Hoffman, 2017). Figure 3 shows both shells.

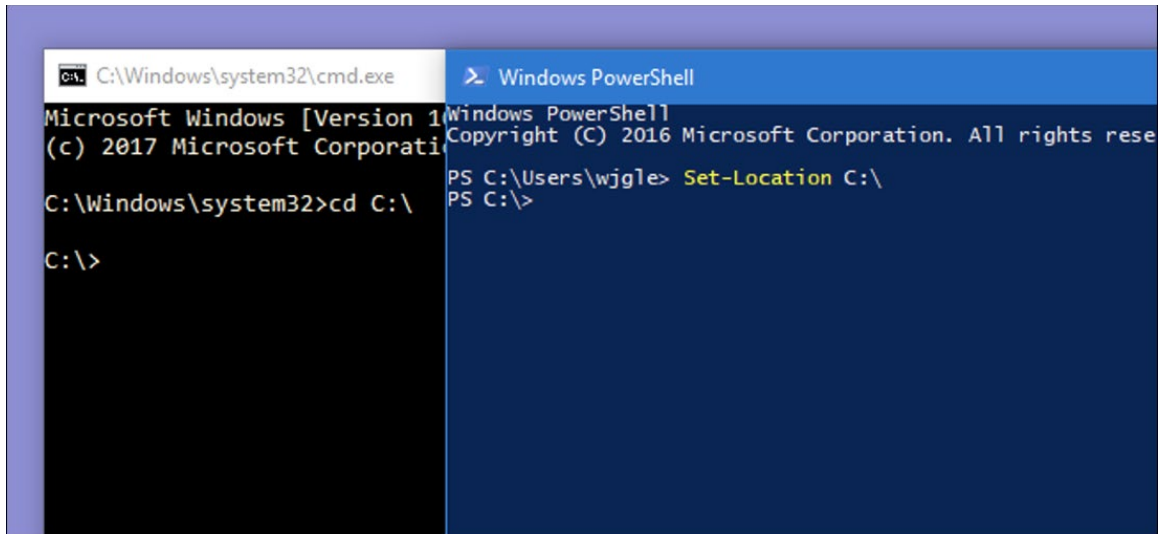


Figure 3. Changing directories in Command Prompt and PowerShell.
Source: Hoffman (2017).

Many computer users are at least familiar with Windows' Command Prompt, and even with a few of the more popular commands such as "ping" or "ipconfig." The command "nmap" with a website as an argument is shown in Figure 4. A journalist at TechGenix promotes at least 11 commands every Admin should know (Posey, 2017). More of these commands should be common knowledge.

```
alexander@alex ~ $ nmap scanme.org
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-19 20:18 PDT
Nmap scan report for scanme.org (74.207.244.221)
Host is up (0.071s latency).
rDNS record for 74.207.244.221: 1186-221.members.linode.com
Not shown: 989 closed ports
PORT      STATE SERVICE
22/tcp    open  ssh
25/tcp    filtered smtp
53/tcp    filtered domain
110/tcp   open  pop3
139/tcp   filtered netbios-ssn
143/tcp   open  imap
445/tcp   filtered microsoft-ds
465/tcp   filtered smtps
587/tcp   filtered submission
993/tcp   open  imaps
995/tcp   open  pop3s

Nmap done: 1 IP address (1 host up) scanned in 3.52 seconds
```

Figure 4. Nmap, a networking command uncommon to normal users.
Source: Wouk (2021).

Using text-based command lines requires memorized knowledge or extensive use of the ‘help’ feature to describe what commands do. Command shells are available on all common operating systems, though possibly named something different. For example, on Unix-based shells like Linux and Mac the shell interface could be called the “Terminal.”

There is a large correlation between Windows commands vs. Unix commands. For instance, the command to display status about a host computer is called ipconfig on Windows and ifconfig on OS X. Other differences come down to a difference in companies, however, and most desired computing results can be crafted from some manipulation of the basic commands on either.

E. CHAPTER SUMMARY

This chapter details several recent cyber-attacks against the U.S. We covered the advantages that a widespread and user-friendly network monitoring tool could bring to workforces that depend on remote traffic in their daily operations. We also discussed the ways in which computer users learn about computer networks and the basic functionality of the command line interface.

THIS PAGE INTENTIONALLY LEFT BLANK

III. NETWORK MONITORING TOOLS

A. INTRODUCTION

Owing to the prevalence of mobile and home network-capable devices, analysis of network packets arriving-to and departing-from one's computer is among the most useful cybersecurity exercises that the public might engage in. Today the Internet is considered a commodity with stricter demands and expectations regarding the quality of its services. There are two distinct communities: a) the user community that includes most of the population who have only a basic understanding of computer networks, and b) the much smaller (by proportion) professional community that enjoys much deeper technical understanding borne of formal education and/or on-the-job experience. The professional community employs tools to understand traffic at a much deeper level; they understand network traffic from endpoint to endpoint, troubleshoot broken connections, and comprehend the operation and consequences of malware. Even today, basic computer networking education is not widespread in the classrooms, so most people learn their skills from their personal experiences. It would, therefore, be beneficial to provide an educational aspect focused on the networking elements for the less-knowledgeable individuals. That would support the growth of their networking literacy, proficiency, and navigation of the networking landscape with enhanced confidence and safety.

Currently, Network Status on Microsoft and some form of Airport Utility on Mac OS devices are used by most people when called upon to examine their personal network. Outside of formal classes or self-tutoring, that type of functionality represents the outer edge of what they know about the inner working of the internet. To address that situation, we propose an application like these programs in the simplicity of the user interface, but that uses the advanced capabilities of the native command shell to provide extra information about network traffic, and about how and with whom their computer is communicating.

B. TOOLS FOR HOME USE

Several attempts have been made to provide users with a baseline knowledge of their networks, albeit minimal in terms of overview and interaction. This section reviews the most notable examples in that category.

1. Microsoft Network Status

Microsoft Network Status is a common view for many home computer users. It is found by clicking an icon in the lower right-hand screen of most Windows taskbars. You can then navigate a few links to change the network settings. While this page displays the most basic settings—like whether the computer in question is connected to the internet or not—none of the shown icons are clickable (i.e., selectable) and there are only a few buttons to either change a simple setting or fix a broken connection. After that, the computer goes through a wizard to perform any functions for the user. Figure 5 shows the status page in Windows 10.

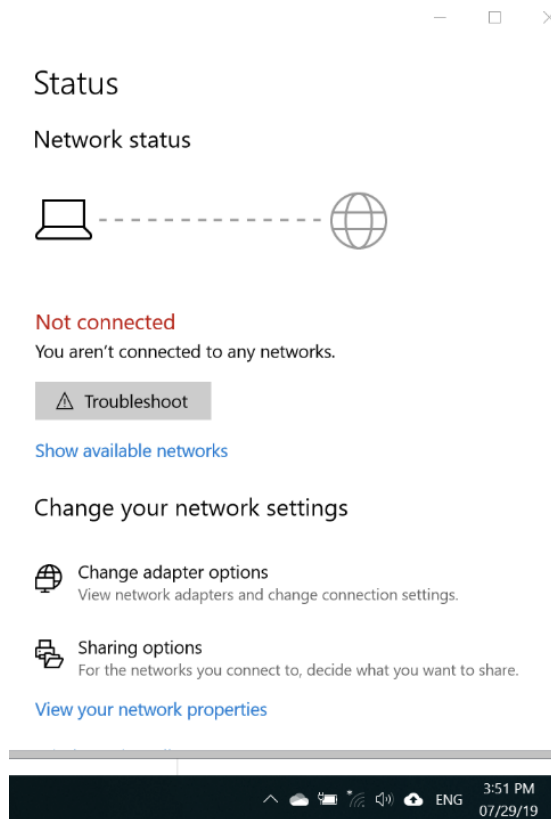


Figure 5. Microsoft Network Status

The computer and network are shown with basic clipart and no color. Some of the items on the page and further into the search could be considered jargon. For instance, the figure above states “change adapter options.” It is highly likely that most people do not even know what an adapter is (as it pertains to a network). Other user interface elements, like a solid line that means ‘connected’ or a dashed line that means ‘broken’, are simple enough. Markedly, this image forms the basis, but also the limit, of many users’ entire networking knowledge.

2. Apple AirPort Utility

When showing the details about their networks, the Apple operating system (MacOS) provides an informative user interface to their users; it presents the network several levels deep, with connected computers as well as routers. Figure 6 shows Apple’s more graphical interface.

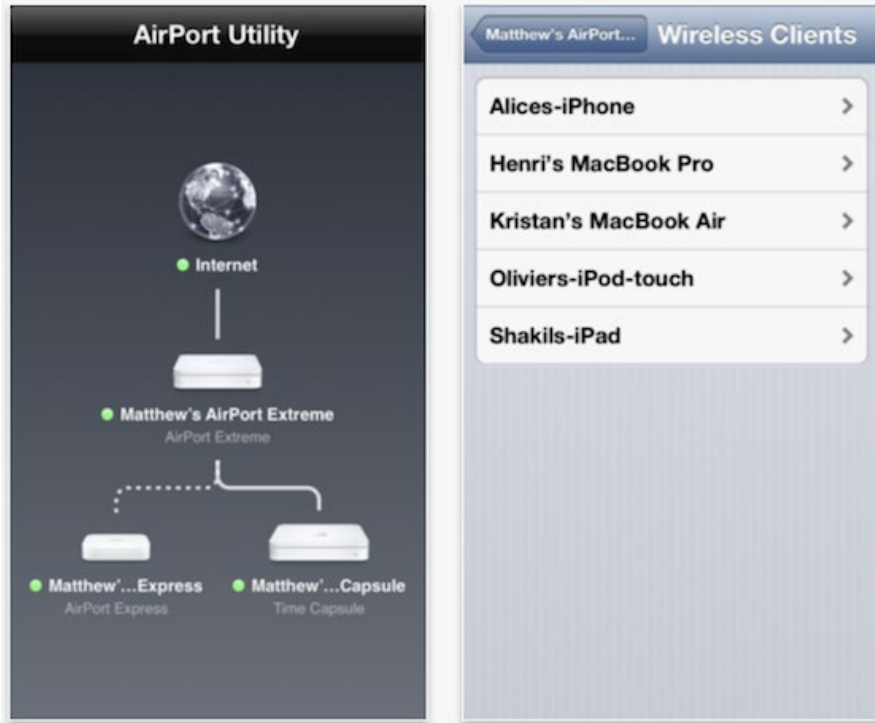


Figure 6. Apple AirPort Utility's cursory yet "pretty" interface.
Source: Lee (2016).

The internet is shown as a three-dimensional (3D) globe, and the background has a gradual fade. Computer icons are shown as 3D objects as well. A green light is shown beside each device to indicate its good connection status. A dashed line shows wireless connections while a solid line shows wired connections.

3. Cross-comparison

The MacOS network display seems more aesthetically pleasing than Microsoft's and it shows more features; that could be an indication that their user design team thought this application would have a higher usage among their base users. This may be because of the shared branding within their ecosystem among their phones, laptops, desktops, capsules, and routers—in comparison to a peer household that might only use Microsoft for its OS only within the user's entire collection of electronic gadgets.

Both systems have access to firewall and virus settings, and both can be used to fix the problems automatically. Neither, however, provides an in-depth operation, such as live network monitoring or access to any devices outside the home system.

C. COMMERCIAL NETWORK MONITORING TOOLS

Business and Open-Source companies have made headway in the presentation of cyberspace as well. Software developers have created applications that can provide the missing networking details to those who need it, albeit specialized and slightly more complicated than native apps developed through home operating systems like Microsoft and Apple.

1. Wireshark (with MaxMind Geolocation)

The Wireshark application's homepage states it is the "world's foremost and widely-used network protocol analyzer" (Wireshark, n.d.). The software monitors incoming and outgoing traffic, and presents it in a text-based, easy-to-read layout that can be paused and scrolled through, analyzed by a user, filtered, and saved. A screenshot of Wireshark is shown in Figure 7. It is a prolific tool used by university students and cyber professionals alike.

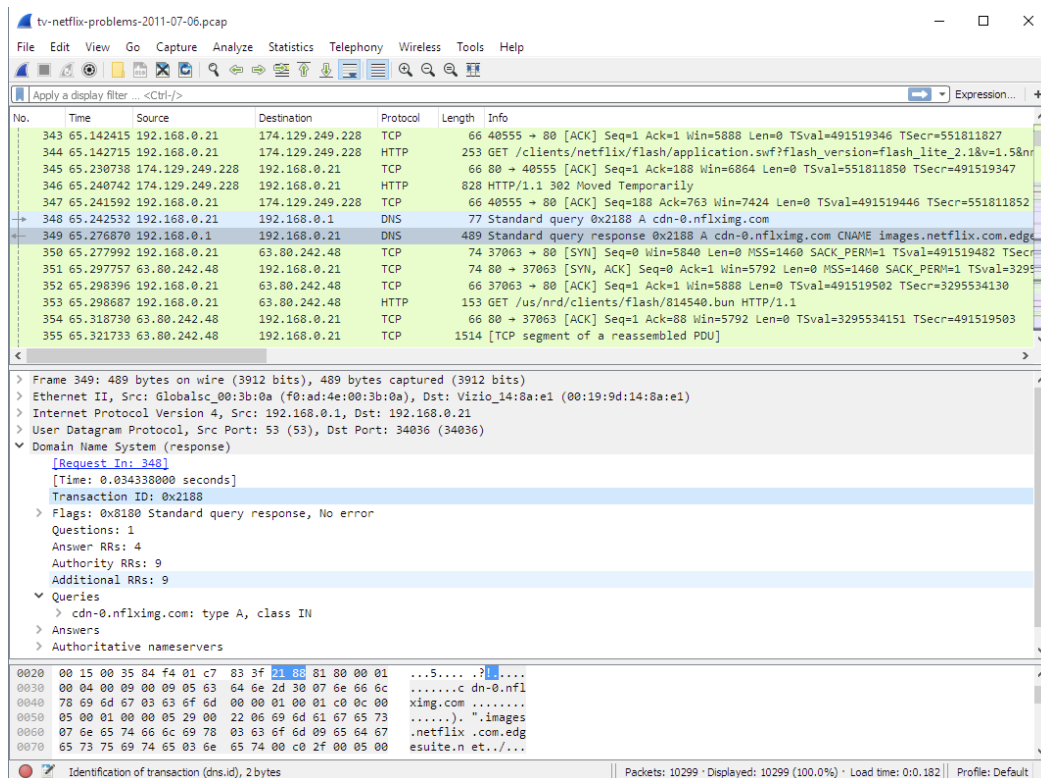


Figure 7. Wireshark showing extensive contents of a full packet.
Source: Wireshark (n.d.).

When using a text-heavy packet capture tool such as Wireshark to comprehend the communication, a user typically reads rows of traffic, looking at the IP addresses to get a sense of ‘the speakers.’ One then turns to the ports and protocols to see what type of dialog is being communicated, and then considers each row (packet) to understand the specifics of the payloads more completely. When looking for abnormalities, the users look for patterns that are indicative of maliciousness.

Undoubtedly, to truly grasp the sought-after traffic requires understanding Wireshark’s particular color-coding and knowing which networking layer is expected to contain whatever specific information is being sought. One also requires much filtering of irrelevant rows to find packets of interest. Therefore, it is not uncommon to have to sift through dozens of rows to find a single indication of improper communication. For those who only want to understand the communication outside of searching for malware, this might include browsing through hundreds of rows of packets. Depending on the sites

open in the browser during the network up-time, that could include ads, frivolous services, and other information.

There are options to filter the information if the IP address of concern is known or if you want to follow a particular protocol exchange like TCP (Transmission Control Protocol). However, to understand the long captures, it takes a considerable manual effort to scroll through hundreds of rows (i.e., multiple screens) of traffic, back and forth, until an idea is refined as to which filters would be most applicable. A module called “MaxMind” can even provide geolocation.

2. TShark

TShark is a well-known network protocol analyzer which allows capturing of data packets from a live network (Chandel, 2020). Figure 8 shows using tshark with the -D command, displaying the computer’s interfaces. It is a text-based version of Wireshark, which in turn is a GUI-version of the common networking command “tcpdump.” Tcpdump is a tool to display traffic from the Command Prompt.

```
root@kali:~# tshark -D
Running as user "root" and group "root". This could be dangerous.
1. eth0
2. lo (Loopback)
3. any
4. nflog
5. nfqueue
6. ciscodump (Cisco remote capture)
7. dpauxmon (DisplayPort AUX channel monitor capture)
8. randpkt (Random packet generator)
9. sdjournal (systemd Journal Export)
10. sshdump (SSH remote capture)
11. udpdump (UDP Listener remote capture)
```

Figure 8. TShark showing computer interfaces.
Source: Kothari (2021).

TShark is based on Tcpdump output, adding some powerful decoders and filters, and written on the console, a stark contrast to Wireshark’s GUI.

3. Network Performance Monitors: SolarWinds, Dynatrace, PRTG Analyzer

Several applications have been made to monitor user traffic. They include intense analysis with tabular charts with extensive headers, graphs in various forms, historical data, mapping, troubleshooting capability, and ability to capture logs. Figures 9, 10, and 11 show screenshots of SolarWinds, Dynatrace, and PRTG Analyzer, representative software showing those characteristics. While the native command prompts and base OS network settings tend to be free, these commercial products tend to cost more which is in line with their advances in features.

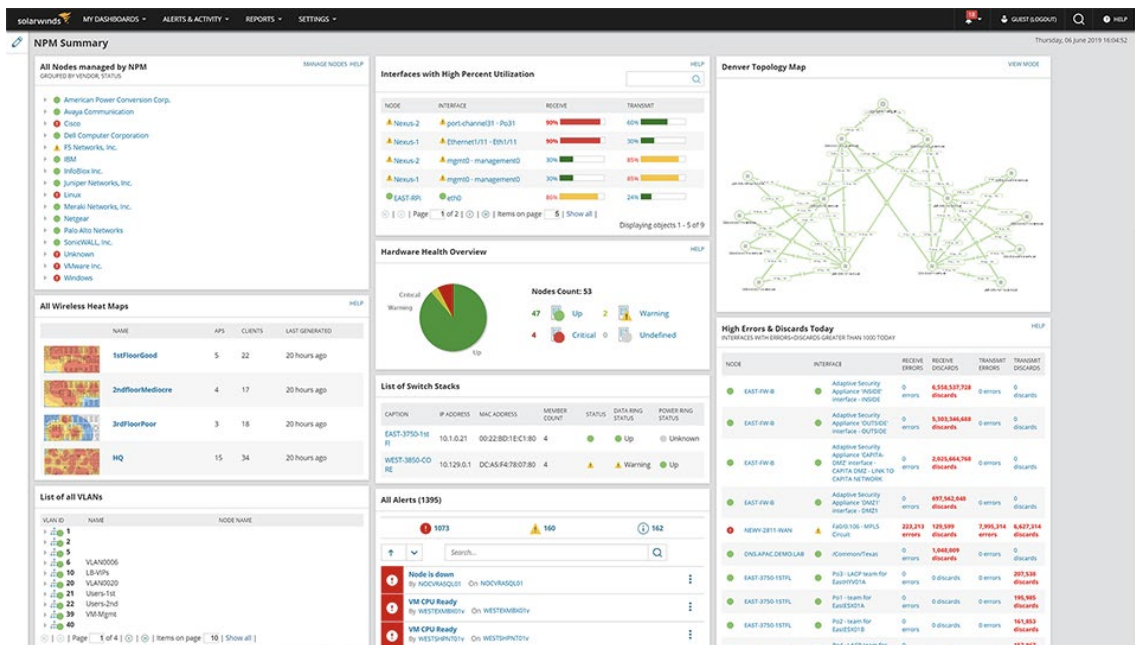


Figure 9. SolarWinds screen showing various networking elements.

Source: SolarWinds (2022).

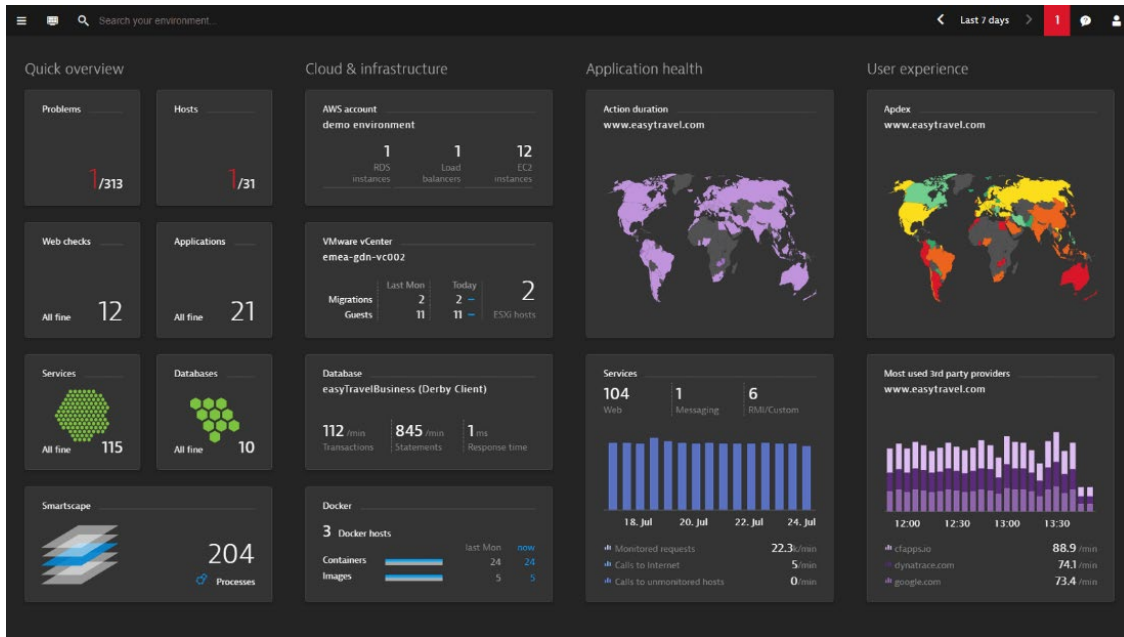


Figure 10. Dynatrace with depictions of cloud and network services.
Source: Dynatrace (2022).

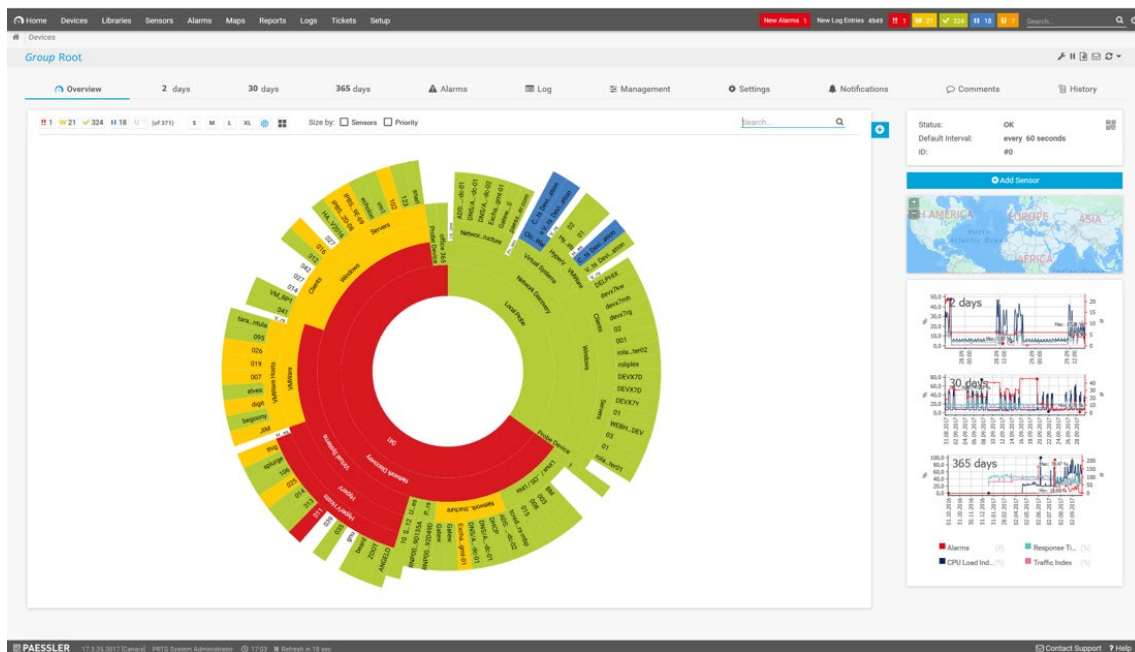


Figure 11. PRTG Analyzer showing worldwide network status.
Source: Paessler (2022).

4. Cross-comparison

Many of these tools fit the requirement for what they were designed for. Command Prompt comes as a native text-input tool, with PowerShell being more advanced. The network analyzers are used extensively by cyber professionals to have easy access to the hundreds or even thousands of computers under their purview. TShark and Wireshark fall somewhere in the middle and can find some usage for almost anyone, ranging from curious home users to hackers to forensic specialists.

D. TOOLS FOR EDUCATION

Many of the varied tools shown thus far have a steep learning curve. So far, this learning is not generally familiar to home users, and information quickly scales up in difficulty as the cyber ladder is progressed. Dedicated education approaches can come in many forms, with some popular ones demonstrated below.

1. Books Dedicated to Networking

Books tend to be the initial introduction to networking that many professionals experience. The specialized topics they teach, such as navigating routing tables and border gateways or calculating checksums, are not generally passed down tribally as the concepts are harder to relay and visualize from a peer to a different coworker.

The downside is that the books tend to be costly and thick, are not as interactive as other methods, and are typically seen as the least engaging learning tool. While the table of contents is helpful, there are no opportunities to quickly find a sought-after topic that modern users are used to, like scrolling or using Control-F to find keywords in the body.

Books are great to record general long-term knowledge. However, the users under consideration will likely need to find their problem in real-time, for instance, if their network goes down or if they think they are being attacked. In that scenario, the book goes from being “less useful” to being “unusable.”

2. University Classes

Relevant classes tend to mainly be attended by cybersecurity students, with even computer science students in other disciplines tending to take the more popular programming or hard-engineering classes. It is well known that cybersecurity jobs have great success in securing applicants who learned their skill by taking much cheaper certifications or bootcamps. These applicants tend to apply much later in life after their initial degree, which is often in something completely different than the cyber domain. Classes also carry the high cost of tuition, and therefore, tend to be more theoretical to legitimize their increased price from cheaper alternatives.

3. Internet Search and Tutorials

YouTube videos and paid sites like Skillshare (Skillshare, 2022) and Udemy (Udemy, 2022) provide excellent tutorials and offer the broad knowledge base to prepare its users for a cyber job, or at least to understand the basics of a home network. Unlike college classes and books, it is easy to browse interesting topics and skip over unnecessary details. One problem is they tend not to offer knowledge as in-depth as books or classes. Also, some of their best video contributors present the material on only a small series on a topic, or present videos that last just a few minutes. Online tutorials can prove to be scattershot in creating competent users on a topic.

4. Cisco Packet Tracer

Learning in diverse environments can be a major benefit to cyber beginners, and one of the best at providing these environments is Cisco Packet Tracer. That application does an excellent job of visualizing a packet's path through a network. A screenshot shown in Figure 12 shows a typical network layout created in Packet Tracer.

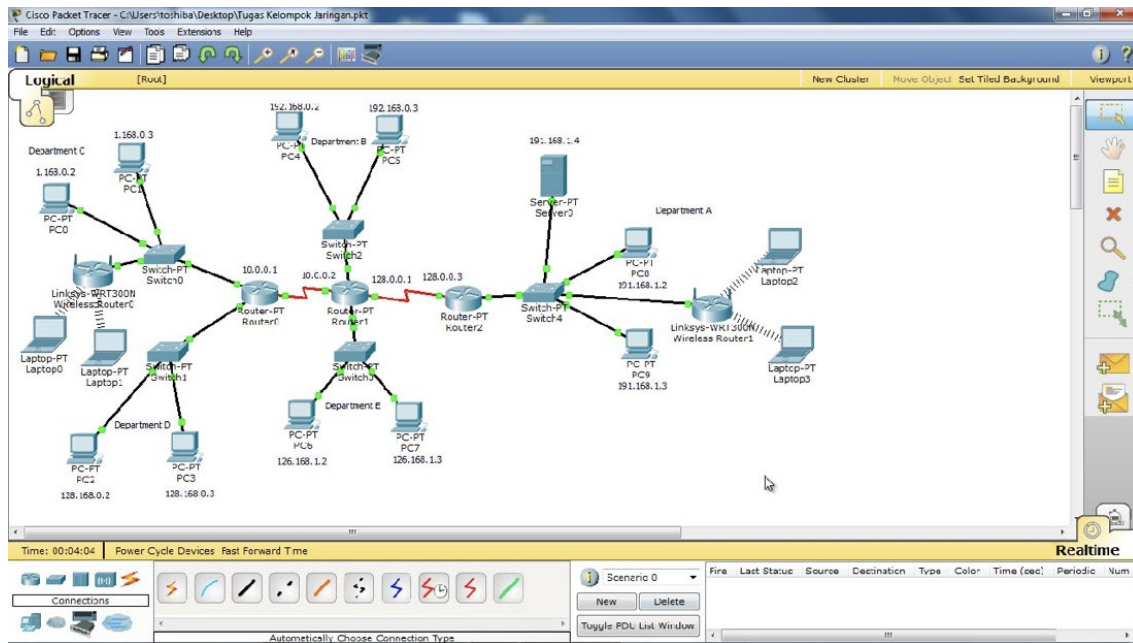


Figure 12. Packet Tracer featuring 4 LANs and their connections.
Source: Cruz Sedy (2015).

While a tool like Packet Tracer is not used outside of a classroom setting, the initial visualization will stay with a user, and the images represented through this tool will be remembered long after the user’s eventual transition to text-based input.

E. CHAPTER SUMMARY

Aside from networking, some tools have been assimilated so well and are so common that their assumed usage becomes standard. For instance, car rental companies assume people with Class C driver’s licenses can drive any of the rental cars, and smartphones are designed such that regardless of their different screen layouts and menu configurations, usually the instruction manual is not consulted. When it comes to networking, there is a place in society for all tools described in this chapter. But there are also significant weaknesses with each that prevent them from making networking command knowledge commonplace among all computer users. The range of currently available solutions is wide, yet imperfect: From slower page-based books to long-duration classes; from expensive commercial products to wordy applications and “user-unfriendly” consoles. Each tool has a certain function in society and can be best-in-class

for a particular audience and a particular situation. However, none of those tools shows a potential of becoming a go-to tool that could be widely adopted by an arbitrary user, and be noted for its simplicity, comprehensiveness, and visual appeal.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. USER INTERFACE DESIGN

A. INTRODUCTION

The basic way in which a user can interact with the operating system is via its text-based terminal called the “command line interface,” or CLI. Popular usage of the interface began around the computer revolution in the mid-1960s and has been a mainstay on all PCs, more than 60 years later. Despite the command prompt’s longevity, the use of the control prompt is not widely employed among all types of computer users, and it is relegated to the system and network administrators. One possible reason for that is the popularity and the ease with which the users can download digital resources from the internet through a GUI. The long history and current usage in CLI basic form without much change are also testaments to how well the command prompt works for the small group of expert users who use it consistently (Campion, 2020).

It may take a large, concerted effort to change the normality of using the command prompt. In the book “Diffusion of Innovation,” Dr. Everett Rogers discusses several factors that can influence the diffusion of innovations like novel ideas, methods, physical objects, or digital systems (Rogers, 2003). Rogers and his research team identified and classified many factors that influence the spread (adoption) of the innovation by a newer audience or the change in habits of the current users to that new system (Rogers, 2003). The factor that reflects the attributes of innovation as they are perceived by the adopters includes the following five elements:

- Relative advantage—the benefits that an innovation brings over the current solution,
- Compatibility—the degree of being consistent with the norm,
- Complexity—the simpler it is to understand and use the innovation, the faster it will be adopted,
- Trialability—incremental, gradual, trials result in easier adoption and higher adoption rates,

- Observability—the results of adoption being visible to other adopters.

In essence, for the large-scale adoption of the user interface, the users' preferences are highly likely to gravitate towards an interface that is close to what is already in use (it does not introduce radically new concepts), simpler than the current solution, while also offering a noticeable advantage, such as speed and ease of use. Therefore, to compete with the long-time incumbent, a new interface must access what makes the old system popular and iterate in a way that makes the system familiar, yet better.

Unwin et. al states that command line interfaces are preferred for three main reasons (Unwin & Hofmann, 1999). Primarily, it is because they provide precision when writing exact commands. Secondly, they provide the ability to repeat prior commands, either by pressing up on the keyboard or by typing “history” to view past results. The final reason is that they can provide easy input when it comes to writing complicated parameters.

Although these benefits are justified, Unwin delineates the drawbacks of the command line, stating that commands can be exceedingly intricate and non-intuitive. Only people who can memorize these complex commands can get the whole value, and even then, they can mistype. Tognazzini (1992) elucidates that these memorization tasks force the user to abandon the cognitive process on the primary task and instead give attention to the secondary task (of command syntax). GUIs can be created to solve some of these problems due to the fact they are easier to learn and allow users to focus on recognition rather than recall. An additional but secondary feature is that GUIs are beneficial for exploratory analysis, and present graphics for those who prefer interacting with images (Computer Hope, 2021). Keeping all these GUI advantages and command line disadvantages in mind, we explore the design of an intuitive GUI to replace the traditional command line.

B. PROTOTYPE IDEATION: PERSONAL CYBER-CONNECTIVITY PICTURE (PCCP)

A well-designed and developed console visualization tool can be used to increase basic knowledge of network analysis terms and their functions and provide a better understanding of machine communication. By iterating on what is useful to know while providing ease of access, our goal is to improve people's comprehension of how computers interact with each other through a network and to educate individuals to become savvier home-users, more qualified job candidates, and more mature novice networking students.

Originally called "Personal Cyber-Connectivity Picture," or PCCP, the GUISE application went through several layout iterations of its user interface. Each iteration improved the delivery of the most important networking information to a novice user in an informational way. Unless a user is an experienced programmer, hunting through the lines of data on analyzers such as Wireshark can be a particularly daunting task. The alternative is to use the native but insufficient OS GUI software, complicated and non-real-time terminal commands, or specialized and costly commercial tools.

Therefore, our design goals are:

1. The user interface will provide fast and efficient probing of the most effective commands on the same interface.
2. The interface will employ the types of user interactions that are simple, easy to use, and familiar to users with no technical expertise.
3. The interface will be capable of providing general education so that a user who is thoroughly familiar with the software would understand fundamental networking concepts.
4. The user interface will provide a clear situational awareness at any point of user interaction: a user will know the options offered via the interface, the way to abandon the current action, and the way to execute a desired course of action.

5. While the interface will be focused on networking commands, its design will support easy extensions to include other groups of commands like forensic commands, file and directory navigation, and pipes and filters.

The basic command prompt does not support probing of unknown commands, does not provide simple interactions, does not have an educational role, nor does it provide easy-to-acquire situational awareness. Therefore, our goal is to take advantage of the basic output of the shell while providing extra functionality to achieve our goals. For this, we must design our prototype with these features in mind from the ground up.

1. PCCP Assorted Applications

An initial idea was to examine extending the functionality of Microsoft’s bare Network Status utility with an ability to open our application upon the click of the represented computer icon, shown in Figure 13. In current forms of Microsoft Windows, the icon is not selectable.

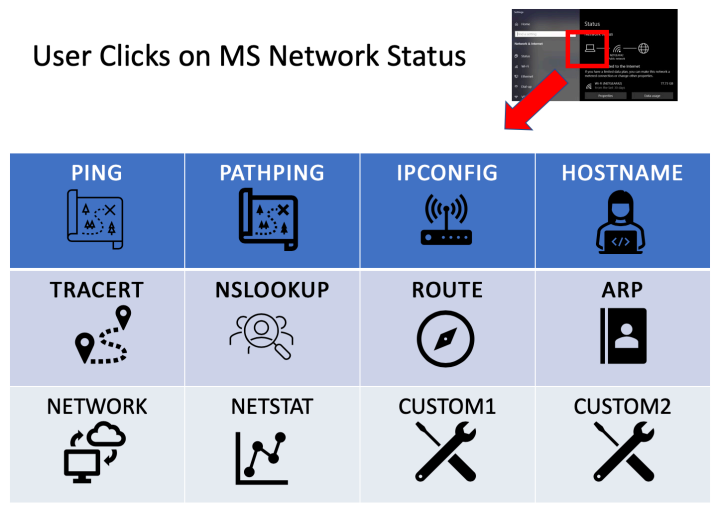


Figure 13. Idea for MS Network Status opening applications graphically

The various menus would have selectable icons themselves, which would open specific software applications or run pre-written commands. For instance, the proposed Network button would open and instantly run Wireshark, while Hostname would show the host name based on the data from running ‘hostname’ in the command prompt.

More sections could have been created, and they would link to various on and off-console applications like Wireshark or off-the shelf commercial software like Dynatrace or PRTG Analyzer, or even native GUI applications like Process Monitor or Task Manager. As shown in Figure 13, most of the written networking requests depended on inputs to the *command prompt*, and thus the decision was made to tighten the focus and highlight those commands, forgoing the approach with components that link to separate applications.

2. PCCP Initial Command Prompt Display

Before the actual development of the interface, the first consideration was to determine how the program would be activated (by clicking the computer icon of MS Network Status) and what would be displayed. Figure 14 shows the initiation of PCCP in the upper left corner.

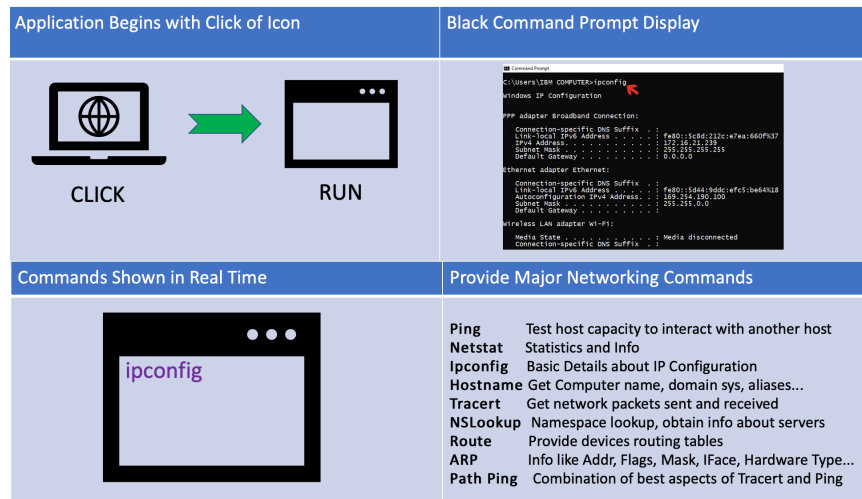


Figure 14. Four Main Requirements of PCCP

The main portion of PCCP, starting at the Home screen, is imagined to look like a standard console. However, this is just a space for visual system feedback and not an interactable shell. Buttons wrap around the console and can be pressed to print text on the screen in real-time, building a working command line, as imagined in the bottom left corner of Figure 15. If the wrong input is selected, that will not produce any result

(through code). That could happen, for instance, by starting a command with an invalid button or by pressing an option button before its associated command is selected.

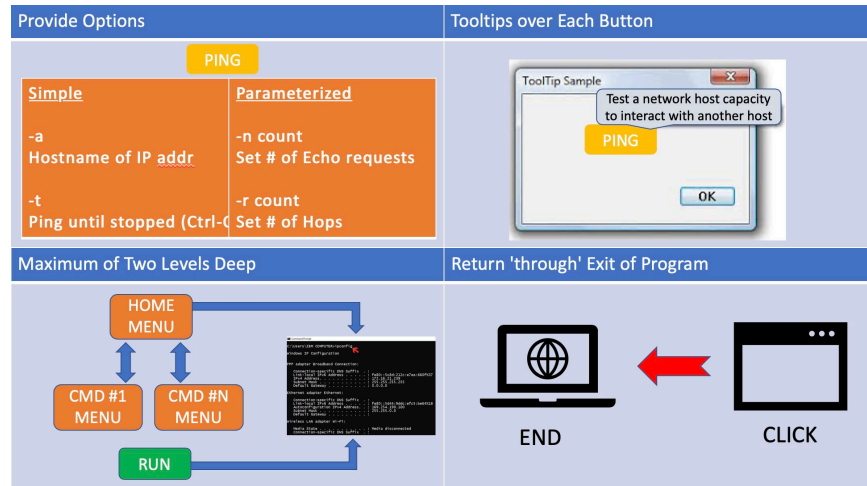


Figure 15. PCCP options, tooltips, flow, and end

As an alternative to allowing (and dealing with) invalid input, one approach would be to grey out any button that cannot be used. However, this would cause constant flickering of buttons (between ‘valid’ and ‘invalid’ colors), even on the correct input. Good design would have much of the screen unchanged, limiting transitions from the prevalent view as much as possible. Therefore, because most of the user input selections are expected to be correct, it is unnecessary to inform a user of mistakes or otherwise put “guard rails” on their input choices.

Starting at the ‘Home’ level, the user sees all available commands. Once they select the desired command, the entire menu would shift to that command’s menu, and a user would be provided with choices specific to that command only. Ideally, there would be buttons for examples, information, usage, and information specific to this command so that a user could feel confident in the usage of the command. As the user selects the elements within this context, that information is held in memory (i.e., pushed onto a stack data structure). Once done, there would be a Return button to go back to the Home menu level, with the display area reserved for the system feedback (i.e., the center screen) continuing to show the text from the command menu. As the user types more commands

this way (going back and forth between the Home menu and some Command menu), the center screen is replaced with the appended chain (commands plus any of its attendant parameters), with the user typing up to “N” commands. After the entire text is entered, consisting of possibly multiple commands chained together, a Run button will allow the user string (of a properly formatted command) to execute. A script reads the text, clears the screen’s content, executes the command, and displays the result back to the screen. If the returned text comes back with more words than the provided display area, a scroll bar provides the option to see more of the lower result. After the command is completed and results displayed, the application will automatically return to the Home screen again, ready to receive commands, and the press of any button would clear the prior information. The screen is cleared because after entering a command, there is a tendency to try a completely different command instead of appending to the old one, and the prior information could be useless. We determined that it is better to have each screen contain only the result of the last command selection rather than have portions of information from prior commands take up screen space.

A peculiarity of the Windows command prompt is that after running a command and seeing a particularly long result in a small window, the text with a desired (selected) command may be out of view. Windows allows simply pressing the up key on the computer to review what was typed. However, considering the educational aspect of our application, constant display of the typed command is instructive, especially for longer commands that are new to the user, so the PCCP leaves the command displayed even after its information is provided, until a new command is entered.

If another command is requested, the screen is first cleared, and the new information is displayed. If the user’s mind changes about the command, there will be a Back button to return from the Console-N menu to the Home screen, offering all command buttons again.

3. PCCP Transition to GUISE

Although the first idea of presenting a network as an easy-to-read and navigable ‘Picture’ is a noble one, it is ultimately outside the scope of this thesis. The second design

iteration of PCCP was easier to implement; it had graphical elements to present and receive text-based results. We realized that there was too much focus explaining the commands by providing a secondary menu (a “Picture”) that was ‘two-levels deep.’ We wanted something simpler than that. The basic remnant of PCCP’s design remained, however, and we created a graphical outer shell for the command prompt that is not as daunting as the bare command prompt and not as complicated as a menu with subsections. Since mimicking a shell command is the basic function of our application we decided to repackage and rename the program “GUI for Shell Entry” with the acronym GUISE. The definition of the word—not the acronym—guise is “a general external appearance; aspect; semblance” (Dictionary.com, 2022), which is the concept of what we desired to accomplish for the shell.

C. DESIGN OF GUISE DISPLAY

Continuing from the basic tenets of our initial ideas, the design of an interface inside the fighter jets developed by the DVI Aviation company has provided additional inspiration in designing an optimal layout for our interface (DVI Aviation, 2022).

A considerable amount of deliberation went into crafting a jet aircraft’s small interior, as shown in Figure 16. Many professionals needed to be engaged in that task. As the experts from the DVI Aviation company suggest, the architecture must be economical with the placement of the various switches and buttons so the pilot can make split-second decisions (DVI Aviation, 2022). Their human factor experts consulted pilots on the best layout for a jet cockpit. It was found that certain characteristics could help define critical controls to assist the pilot in efficient recognition of where each item in the cockpit was. The company suggested the following guidelines for the elements of that user interface:



Figure 16. Jet fighter cockpit interior
Source: World War Wings (2022).

- **Control Shape.** Shape was used to differentiate the purpose of the controls, with tactile cues indicating the control's purpose. They suggested the use of an airfoil or wing shape to control the aircraft's flaps or a miniature wheel for the landing gear retraction.
- **Control Size.** Controls of different sizes also allow differentiation. That, however, provides less advantage than shape changes because the distinguishable difference in size might not be identified as quickly.
- **Location.** Controls should be separated by enough distance to distinguish each element of the user interface.
- **Colors.** The colors red, orange, yellow, green, and blue are easy to discern by a human operator. That understanding drove the selection of the final set of colors in the user interface. For instance, improper/dangerous operation and warning lights were red or amber, and a green color signified proper operation or the position of the landing gear.

- **Mode of Operation.** The “way” in which the controls operate can also help categorization. For instance, when comparing two controls like a push-pull type vs. a rotary type, though they might have similarities in colors, sizes, shapes, etc., they will still be very distinguishable because they operate so differently.
- **Labeling.** Names should be clearly labeled and unambiguous.

While not all these characteristics are used in this thesis, the DVI list can guide the construction of a usable computer interface. We can develop a menu with easily recognizable elements by using coloring schemes that contrast and make sense, varying the button sizes based on function, and grouping buttons far apart.

The depicted flight simulator in Figure 17 shows many of the highlights mentioned in DVI’s layout suggestions. Although the entire interior of a jet plane was not used on the in-game screen, the basic features are displayed prominently (i.e., distancing of the buttons, the type of operation between the rotary, single push and dual-push buttons, their sizes, and the color contrast on and off-screen.) The basic design of our final prototype is based on these tried-and-tested elements.

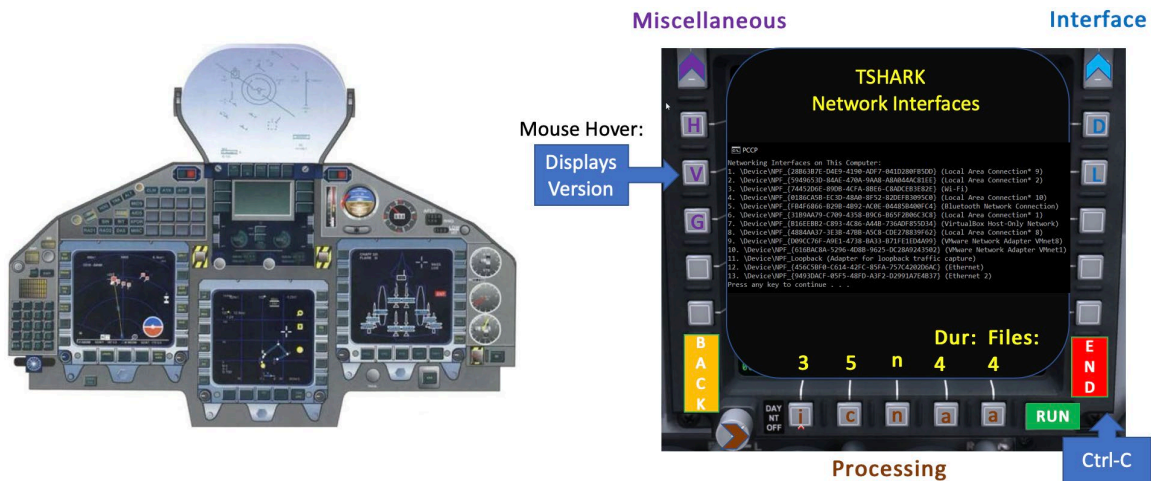


Figure 17. Flight simulator in-game screen vs. initial GUISE layout
Source: Kellerer et al. (2011).

D. GUISE LAYOUT

The GUISE layout consists of menu buttons for commands positioned at the top, services (on the left), single options (on the right), and parameterized options (at the bottom). Arrow buttons are at either end of the menu to allow scrolling through each, and a descriptive label provides the menu's name. The action buttons—'Run', 'End', and 'Back'—are in isolated locations and have a color that contrasts with the other Menu buttons. Several earlier versions are shown in Figures 18, 19, 20, and 21, with the final blueprint shown in Figure 22.



Figure 18. Early PCCP Mockup #1: Showing depressed (grey) buttons, and how simple / parameter options and tooltips work

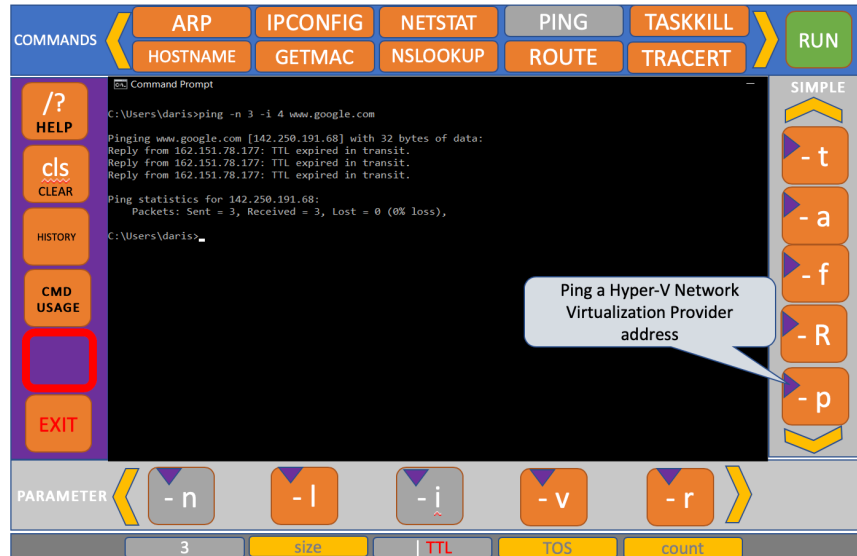


Figure 19. PCCP Mockup #2: Showing tooltips (in purple) on options, replacing separate “usage” button



Figure 20. PCCP Mockup #3: Showing InputField to replace parameter buttons, and TargetName to input strings

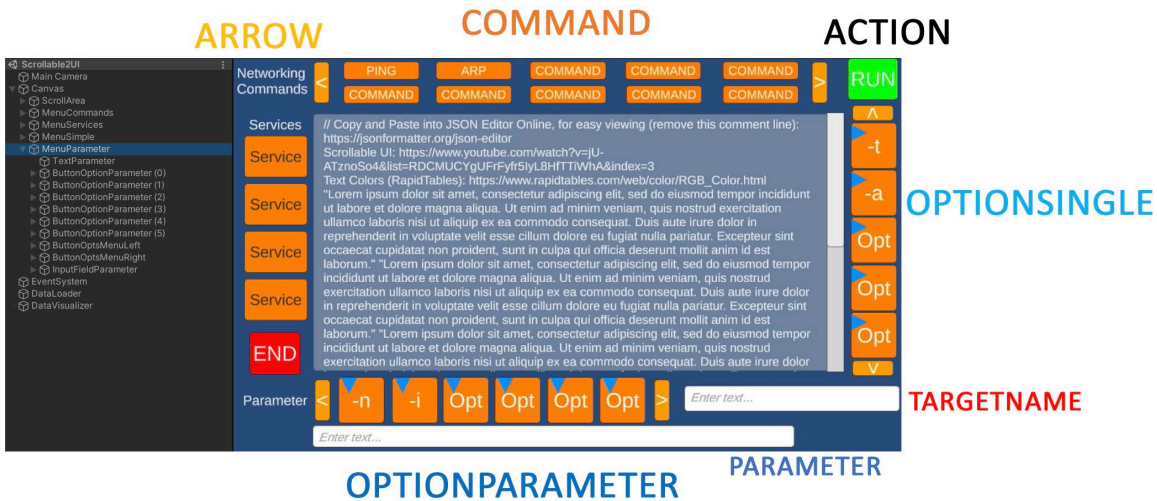


Figure 21. Later PCCP version with newer tooltips, arrow bars, TargetName, and InputField

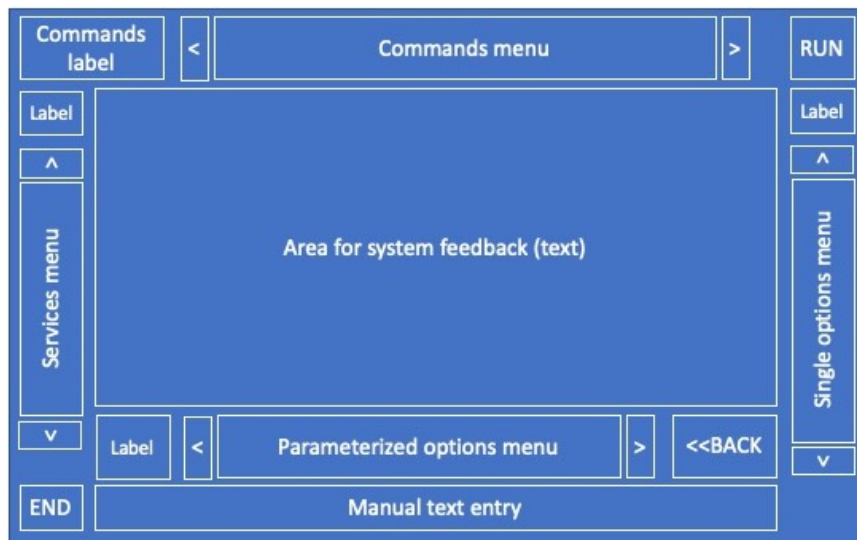


Figure 22. GUISE blueprint

E. GUISE MENUS AND USER INTERACTIONS

1. Commands Menu

The hub of the application, the “Commands Menu,” consists of ten buttons along the top of the GUI. Catering to the top networking commands would provide the greatest chance of being useful for a user, thus allowing for the most practice. Once these are verified to work, the system can be designed to be customizable, providing a method to

add any arbitrary command with any option. Between consulting the suggested popular networking commands from EDUCBA (Swade, 2020), TechGENIX (Posey, 2017), and Microsoft Documentation (Microsoft | Docs, 2022), we settled on the following ten baseline commands:

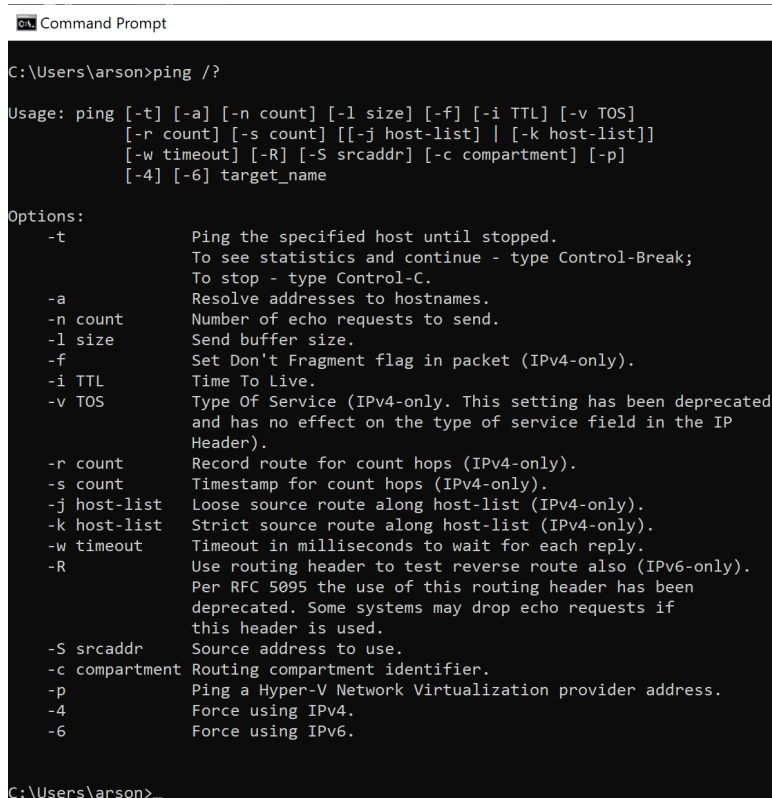
- ping Test host capacity to interact with another host
- tracert Gets network packets sent and received
- arp Info like address, flags, mask, interface, hardware, etc.
- ipconfig Basic details about Internet Protocol (IP) configuration
- whoami Displays users, groups, and privileges information
- route Provide devices routing tables
- hostname Gets computer name, domain system, aliases, etc.
- nslookup Namespace lookup, obtain info about servers
- net Manage and configure the OS from the command line
- netstat Statistics and info

The label used for this part of the menu is called “Networking Commands.” Ten corresponding buttons are displayed together as a Networking “Group.” Arrow buttons were created to the left and right of these groups to provide reach and rotate through different groups. Other groups that were considered as the extension of the basic set of networking commands were “Forensics” (commands to investigate a drive for malware or unanticipated events), “System” (for hardware and networking artifacts), and “Processes” (to display and interact with running programs). The Networking Commands group is the only one that was fully implemented, and implementation of other groups of commands was outside the scope of this thesis.

2. Services Menu

Services are augmentations to standard commands that can provide extended information or additional complexity outside the purpose of the command. These include help and find features, or showing the directory or processes, among others.

The help menu can be shown at the terminal by typing a “-h,” “--help,” or “/?” just after any command, as Figure 23 shows for the *ping* command. Sometimes just typing the command with no input and pressing enter can get to the help screen. Help can be used to show more details on how to use any command.



```
C:\Users\arson>ping /?

Usage: ping [-t] [-a] [-n count] [-l size] [-f] [-i TTL] [-v TOS]
           [-r count] [-s count] [[-j host-list] | [-k host-list]]
           [-w timeout] [-R] [-S srcaddr] [-c compartment] [-p]
           [-4] [-6] target_name

Options:
  -t           Ping the specified host until stopped.
               To see statistics and continue - type Control-Break;
               To stop - type Control-C.
  -a           Resolve addresses to hostnames.
  -n count     Number of echo requests to send.
  -l size      Send buffer size.
  -f           Set Don't Fragment flag in packet (IPv4-only).
  -i TTL       Time To Live.
  -v TOS       Type Of Service (IPv4-only. This setting has been deprecated
               and has no effect on the type of service field in the IP
               Header).
  -r count     Record route for count hops (IPv4-only).
  -s count     Timestamp for count hops (IPv4-only).
  -j host-list Loose source route along host-list (IPv4-only).
  -k host-list Strict source route along host-list (IPv4-only).
  -w timeout   Timeout in milliseconds to wait for each reply.
  -R           Use routing header to test reverse route also (IPv6-only).
               Per RFC 5095 the use of this routing header has been
               deprecated. Some systems may drop echo requests if
               this header is used.
  -S srcaddr   Source address to use.
  -c compartment Routing compartment identifier.
  -p           Ping a Hyper-V Network Virtualization provider address.
  -4           Force using IPv4.
  -6           Force using IPv6.
```

Figure 23. Ping using /? to show its help menu, options, and their descriptions

3. Simple and Parameterized Options Menus

Some commands, like *ipconfig* or *hostname*, may take no options (no arguments) and simply provide the status of your computer. Other commands like *ping* need

arguments, several of which are shown in Figure 23. The *ping* command interacts with devices outside of the home router, and needs a name, an IP address, or a location finder of some kind to establish a connection with the outside. Options generally come in two forms: a single option itself (usually a dash followed by a letter or word), or a parameterized option, which is an option (a dash and a letter or word) followed by an argument belonging to the option. The extra information gives the option more context. An example of a simple option is “-a,” while the parameterized “-n count” demonstrates that the `-n` option (n standing for number) would need to be followed by an integer (count) representing the number of echo requests—pings—to send.

Most commands can generally chain several options together to form a much longer and more specific filter. One example of an extended command using multiple types of options would be “*ping -n 3 -i 5 www.google.com.*” Upon review of the descriptions seen in Figure 23, we see that this would result in ‘pings’ of the Google Web server with three echo requests, with a time to live of five (seconds) each.

In creating the data used in GUISE, the exact descriptions from the help menu were copied and pasted into the tooltips and descriptions of the software. However, using the above help features was not always available; for instance, the *perfmon* command has no help feature. When the native help menu was not clear or present, we found documentation online (*perfmon*’s description was taken from Microsoft’s documentation).

4. Manual Text Entry

Many commands finally end with a target of some sort, like an IP address. To support the manual entry of the text like an IP address, GUISE includes a long bar across the bottom of the interface. This capability is useful when looking for a website, inputting a numeric internet address, or typing the name of a document with its extension. While simple applications could be created that only consisted of button presses or drag and drop actions, the most useful commands have arguments for arbitrary text, and the projected users of our application will need this feature.

The menu labels with white text are provided near their respective buttons. The label for the group of main commands is emphasized—its text is colored yellow, and it is in a bigger font. The alternating arrow buttons are presented adjacent to each group of buttons to allow reach and rotation through additional groups of the same type of element (‘paging through’).

5. Tooltips

Tooltips are created by making a Unity sprite (a triangular image) as a UI button and placing it in the upper left-hand corner of each menu button, as a child. Each has a script component with fields called “header” and “content,” which respectively provide a readable name and description of the parent button.

F. VISUALIZATION OF GUI ELEMENTS

The prototype GUI consists of a replica console in blue (mimicking PowerShell), with various buttons surrounding it that use scripts to imitate typing arbitrary commands on a console.

1. Unity as a Development Environment

Unity is one of the world’s leading cross-platform gaming engines (Unity Technologies, 2022a). It is used to perform modeling in graphics-focused applications, from simple 2D art and 3D interactive environments to real-time photorealistic movies. Unity was selected to support user interaction and display graphical elements because of its simple interface, wide choice of coloring schemes, fast rendering (requiring almost no compilation time for the simple images and buttons we require), free usage, and cross-platform support.

2. GUISE Features

Several features were selected as a general approach for the GUI:

- Black/Blue command prompt display
- Commands and options shown in real time as typed

- Provide major networking commands
- Provide options
- “Single” click for individual options
- “Settings” to type in values for numbers or strings
- Tooltips over each button
- Maximum of two levels deep
- Return ‘through’ exit of program
- Optional: Custom button, advanced vs. beginner settings

3. Options Layout

Command scenes have options surrounding the main screen to allow building lengthier and more complicated commands. A full list of available options can be shown by visiting any command’s help menu. The two types of buttons for the options, called “Single” and “Parameterized,” are distinguished by location; they have the same shape since they basically accomplish the same thing. The parameterized buttons are placed along the bottom, with an area for a manual text input below that group. Tooltips attached to each option describe their features in detail.

4. Color Scheme and Sizing

In complying with the Jet Cockpit methodology mentioned earlier in this section, we selected standard colors used in other software applications. The “*Main*” groups of buttons consist of Commands, Services, Single options, and Parameterized options; each menu option is represented as an orange button with corresponding text distinguishable from other elements of the GUI. “*Action*” buttons have uniquely distinct colors. The blue-shaded ‘Back’ (backspace) button, which removes characters, is visibly contrasted from the Main buttons. The ‘Run’ button, which runs the commands on the screen, is green while red is used for ending the program with the ‘End’ button. The Arrow button, used

to simply rotate through pages of commands and options is stylized in plain white with black characters. The screen background is created in matte blue, and the overall background is dark blue, to be complementary with the Microsoft’s currently recommended command prompt, PowerShell. Representative buttons are shown in Figure 24, and the exact sizes and RGB color schemes are shown in Table 1.

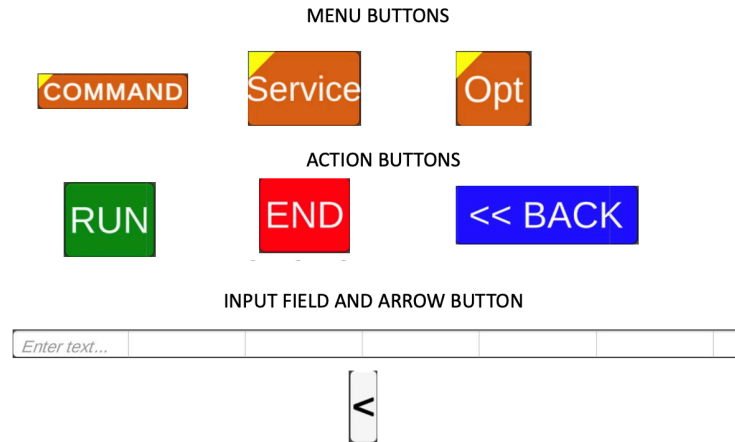


Figure 24. Color scheme of GUISE buttons

Unity’s “autosize” feature allowed us to constrain the text font size between 10 and 24, which allowed smaller strings like “-a” to be seen clearly, and large strings like “/fo table” to shrink and fit within the button surface. Anything larger than that was truncated or abbreviated.

Table 1. GUISE sizes and colors

Element	Width x Height (units)	RGB Color
Command Button	100 x 25	210, 105, 30
Service Button	75 x 50	210, 105, 30
Option Button	50 x 50	210, 105, 30
(Up) Arrow Button	20 x 50	255, 255, 255

Element	Width x Height (units)	RGB Color
Run Button	60 x 50	35, 140, 35
End Button	60 x 50	255, 0, 0
Input Field	625 x 30	255, 255, 255

Sizing was provided in integer increments, as seen in Figure 25. The formation was ensured to be standard by isolating a single menu group, then setting its first element to position (x: 0, y: 0). Sizes of other buttons were adjusted based on this initial reference. Five spaces of margin were given between elements within menus and from the edge of the canvas or scroll area. Afterward, the edge of the scroll area within the canvas was calculated based on the total distance traveled of the margins plus the sides of the menu buttons.



Figure 25. Button positioning based on Command 0

5. GUISE GUI Layout and Operation

The layout of the GUI, as illustrated in Figure 26, included the middle scroll area (the effective command prompt reserved for system feedback) equivalent to the width and height of the native command prompt upon opening. The excess margins were added around the edges of the space in the middle to accommodate various menus and their buttons. The entire GUI takes up about 1/2 of the width and height of a 15” laptop screen, which allows for legible text and easy selection of each GUI element.



Figure 26. Final GUISE interface

The application opens at a terminal prompt called “GUISE >.” A user can begin selecting commands by pressing one of the top Command buttons and if desired, keep adding Single Options (seen on the right) to make a more complex statement. Alternatively, one of the Parameter Options can instead be selected (if available for that command); however, it must be followed by entering an argument in the text input field below it. The field provides its ‘parameter’. Unity’s input field allows it to be registered by either pressing the Enter key or clicking anywhere within the application with the mouse. This way, long and complicated statements can be created from a single command and several single and parameterized options. Services from the left menu can ‘chain’ complicated statements by using pipes or redirections, just like in an ordinary command line. Alternatively, a command can be selected, followed by the ‘Help’ command (from the Services menu) and then the ‘Run’ button right after. Figure 27 shows an example of running the system after typing in the ping command with several arguments. The system feedback is the text that shows the command prompt’s help features.

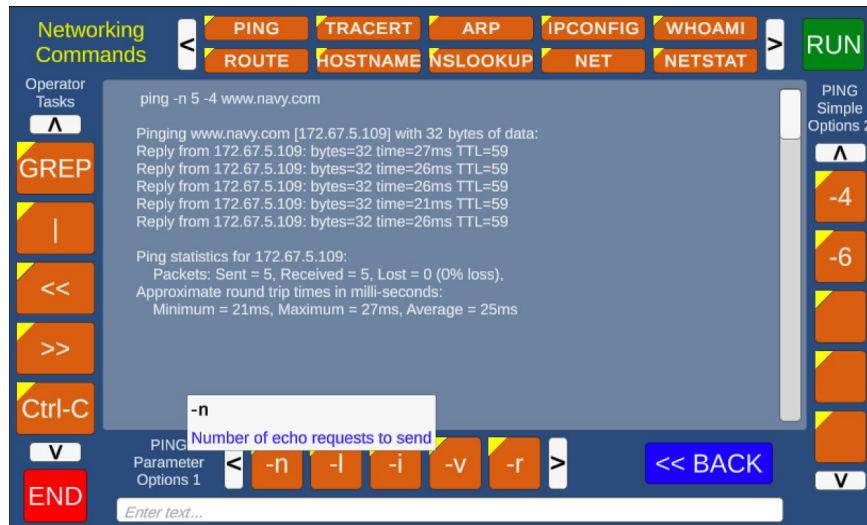


Figure 27. GUISE after running a command, hovering over an option

If an error is made while selecting different buttons from any menu, the ‘Back’ button can be pressed to delete the last user entry. For instance, if the user sequentially clicks “*ping -a*,” selecting the ‘Back’ button once would remove the ‘-a’ element, and clicking it again will remove the entire word ‘ping.’ Once the desired command and its options are all selected, pressing the ‘Run’ button will execute the command and the results are shown on the screen. The application is then ready to receive the next command. Clicking any button clears the screen and starts the process over.

Finally, GUISE’s most powerful feature comes in the form of tooltips. Each menu button has a smaller section in the upper left-hand corner that provides context for the underlying command. Providing instantaneous information is helpful in learning about hitherto unknown commands, providing education, or saving time when uncertain of a command’s operation, thus increasing performance. Currently we only provide the name and description, but additional content can swiftly be added. Additional content could include specific examples or demonstrating the structure of how to write a complex command. Abiding by Fitts’s Law (Fitts, 1954), we made the tooltip large enough to be easily found and referenced, but small enough to ensure they would not distract users during normal usage if tooltip consultation was not needed.

G. CHAPTER SUMMARY

This chapter provided details about the initial prototype of GUISE, originally called Personal Cyber-Connectivity Picture (PCCP), the follow-on design of GUISE based on a fighter pilot layout, and the several iterations GUISE has undergone. We provided specifics of the layout of GUISE including the purposes of the various menus, their placement, sizes, shapes, and colors, and explained the basic operation of entering commands and their buttons' tooltips.

THIS PAGE INTENTIONALLY LEFT BLANK

V. SYSTEMS INTEGRATION AND IMPLEMENTATION

A. INTRODUCTION

Once the design features of the interface were determined, we proceeded with the prototype development. The final product has six C# scripts, one text file for the output, and one JavaScript Object Notation file for data. While the inclusion of audio could be considered to make the application more multimodal, that was not a pertinent goal of our work, and it was not added. The inclusion of audio for button clicks and swiping sounds is more suited for games, like Solitaire, and is typically not supported in professional applications like Word, Excel, or other production software. The final GUISE application GUI was shown in the prior section, in Figure 26.

B. GUISE C# DEVELOPMENT IN UNITY

The first task was to create a GUISE interface design (the layout of all visual elements) in a Unity ‘Scene’, which is the environment that all the interactive application elements operate in. All visual elements like buttons and text are placed on a UI Canvas, with simple drag-and-drop actions. The Scroll Area, which mimics the console, was the first item to be placed in the center.

The next part of the prototype consisted of making the interface interactive. The TextMeshPro module is used to format the text; that is the newer version of Unity’s legacy UI Text system, with advanced text rendering and better styling and texturing (Unity Technologies, 2022b). The buttons and the scroll area all contain this new rendering system to make the text within them appear sharper and more visually appealing. The “Menus” system has ten UI TextMeshPro buttons at the top for the commands, five on the left side for the Services, five on the right side for the simple options, and five across the bottom for the parameterized options. There is an elongated input field underneath the parameters for specifying values like internet addresses, numbers, files, or processes. Eight buttons for arrows are placed adjacent to each series of menu buttons, and respective labels are placed nearby. The “action” buttons for ‘Run’, ‘End’, and ‘Back’ are placed opposite each other with unique colors. Finally, smaller

yellow triangular buttons are placed within each of the menu buttons, facing the upper left corner, to provide the tooltip functionality for each button. The layout shown in Unity, with the hierarchy of UI elements on the left side and before running, is shown in Figure 28. An example of navigating to a TextMeshPro UI button is also shown.

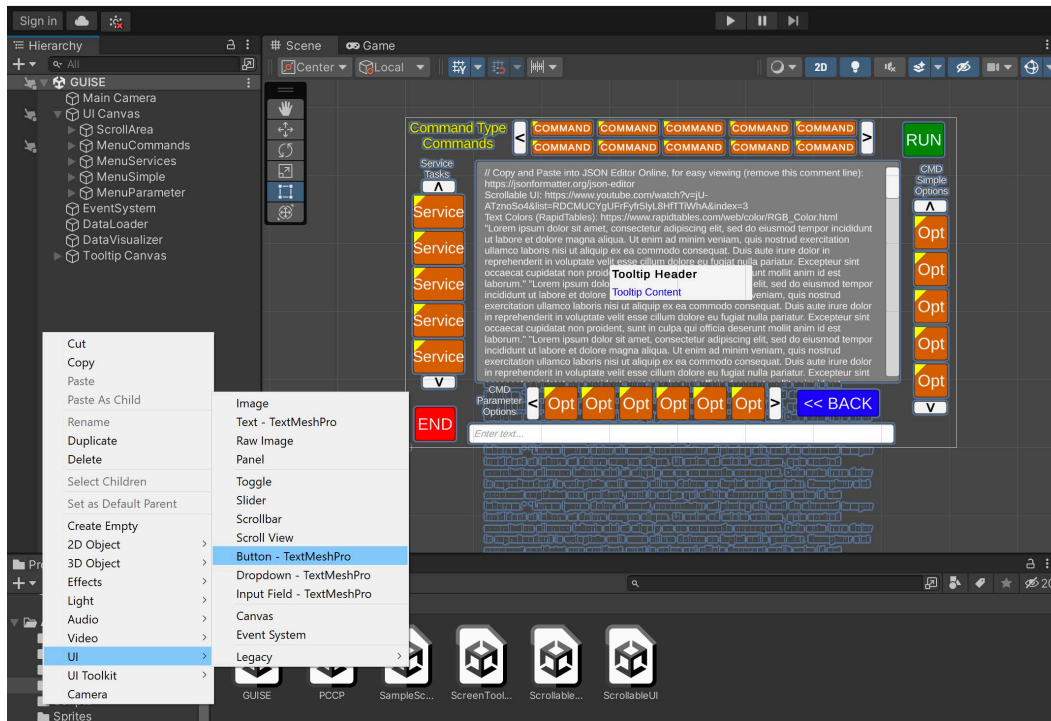


Figure 28. GUISE UI elements within the Unity game engine

C. NAMING CONVENTIONS

In order to support the overall structure of the interface, the names, elements, and functions are created in the practical programming method of first labeling the name with the type (e.g., “button,” “inputField”) or action (e.g., “rotate,” “set”). Other best programming practices were taken from standard C# coding paradigms (Avantgarde Software, 2019) such as using camelCase, access modifiers like private and public, and placing comments above every method, among many others as shown in Figure 29.



Figure 29. Expanded menu (left) and collapsed method conventions (right)

D. JSON DATA

Once the interface layout was complete, the meta data set was copied into a JavaScript Object Notation (JSON) file, as shown in Figure 30 and Figure 31. That data set consisted of the textual descriptions for each command, the characters for options and parameters, and the descriptions of each option. The information was gathered from various sources, the main one being the help menu from the Windows command prompt. If sufficient information was still missing, the necessary data were gathered from docs.microsoft.com as a secondary source. The author provided clarifications in very few instances, for example, if the command's definition was common knowledge or there was a better formulation online. Approximately 1000 lines of JSON data were manually typed in the end.

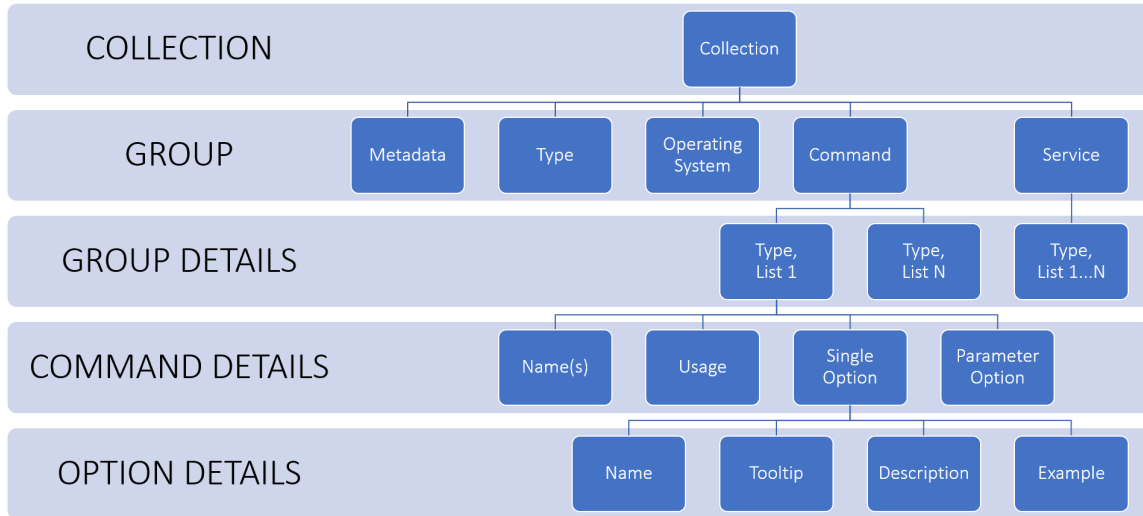


Figure 30. Overview of JSON Fields

```

commands_networking_collection.json
Schema: <No Schema Selected>
1  {
2  "type": "CommandsCollection",
3  "metadata": {
4  "url": "https://techgenix.com/top-11-networking-commands/",
5  "title": "CLI Commands",
6  "count": 11
7  },
8  "os": "Windows",
9  "serviceGroups": [ ... ],
146 "commandGroups": [
147 {
148 "commandType": "Networking",
149 "commands": [
150 {
151 "commandName": "ping",
152 "fullName": "Ping",
153 "shortName": "ping",
154 "buttonName": "PING",
155 "usage": "Verifies IP-level connectivity to another TCP/IP computer",
156 "optionsSingle": [
157 {
158 "optionName": "-t",
159 "tooltip": "Ping the specified host until stopped.",
160 "description": "Ping the specified host until stopped.\nTo see",
161 "example": "ping -t 8.8.8.8 ...ping google"
162 },
163 {
164 "optionName": "-a",
165 "tooltip": "Resolve addresses to hostnames.",

```

Figure 31. Expanded JSON data showing fields for commands and options

E. DATALOADER AND DATAVISUALIZER SCRIPTS

The Dataloader and Datavisualizer scripts provide the main way in which the application is run. Dataloader has classes that match every heading and subheading in the

JSON data. When the application is started, Dataloader imports the entire JSON information into a single in-script variable called “data.” The script then calls Datavisualizer with this data as an argument. If one understands the classes as a programming concept, simple programming will easily allow adding more data to the JSON and, as a result, have more command groups. Combined with the ability to scroll through each command group with the rotational arrow buttons, these features allow the GUISE application to be easily extendible.

Various methods inside Datavisualizer initiate system reactions (system feedback) when buttons are pressed. Although about 40 methods were needed to complete the full functionality of the GUISE interface, they are usually grouped by their similar operations in four groups of four—one for each menu type. In Table 2, “assign<Menu>“ means Menu is a stand-in for the four types of Menus: assignCommands, assignServices, assignSingleOptions, and assignParameterOptions. All methods are described in Table 2:

Table 2. Datavisualizer methods and their descriptions

Method Name	Description
loadJSON	Loader calls this 1 st , provides data
assign<Menu>	Applies an entire group’s text on buttons
assign<Menu>Tooltips	Applies an entire group’s tooltips
assign<Menu>ButtonName	Helper applies text to an individual button
assign<Menu>ButtonNameTooltips	Helper for tooltip’s header & content
resetScrollArea	Erases console information
writeScrollArea	Applies console text
getCommandString	Get global variable commandString value
backspace	Delete entire last element (e.g., “ping”)
setRunText	Applies result of a command to console

Method Name	Description
clearFile	Erases (prior) output.txt content
set<Menu>Text	Applies the words on a button to console
storeStatus	Saves current Group, Index, & Page
rotate<Command/Service>Groups	Arrow button slides entire group left/right
rotate<Menu>	Arrow button slides single page left/right
executeCommand	System returns results of user's command
writeString	Inputs results of command to output.txt
readInputString	Reads user's typed input and clears field

F. STATUSNODES: CONTEXT FOR INDEXES, PAGES, AND GROUPS

If a Command is selected—for example, command-1—its Single and Parameter options are shown around the right and bottom edges. However, if instead of selecting a Single or Parameter option for command-1, a user selects another command—i.e., selects command-2— then those Single or Parameter buttons would no longer be applicable to that Command. Therefore, both option menus are switched to represent the Single or Parameter buttons associated with the new Command. This entire state of the screen—the Services shown on the left, and the selected Command shown on the top with its Single and Parameterized options around the right and bottom edge—is called a “context.” The StatusNode class was created to save the contexts, indexes, pages, and groups associated with the buttons of a scene, as seen in Figure 32. We introduce and explain each term in sections that follow.

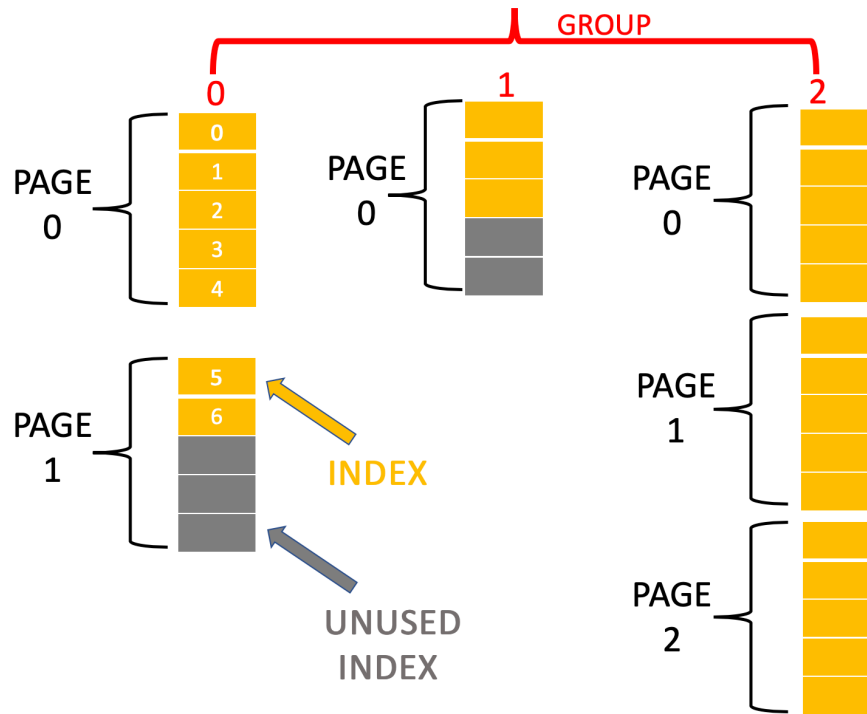


Figure 32. StatusNode Layout showing Groups, Pages within Groups, and Indexes within Pages

1. Indexes

Each menu button has an index that is associated with it. The *ping* command is the first in the list in the JSON data set, and so it has Command Index value of 0 (there are ten commands in one set). Service Index value of 0 (out of 15 total services) relates to the *grep* (global regular expression print) service, and so on. To describe a certain element, one would have to know where it is indexed in the JSON data set. Afterward, the assigned function uses that value to find it in the Datavisualizer’s data variable and replaces the button’s value as characterized in the JSON database.

2. Pages

Menu buttons come in groups of five (or ten for the Command menu), starting with Index 0, so all index values between 0 and 4 would be shown together. If an arrow button is clicked, we “slide” buttons 0 through 4 off screen and show the items with index values 5 through 9. This association of a set of indexes is called a *Page*. While we

do not necessarily need to change the numbering of indexes, it helps identify the exact button that is activated.

All buttons are labeled from the initial interface with a certain unchanging number. The upper Single Option button has a *button* index value of 0, and the upper right Command button has index value of 4 (since it is the 4th one from the top, going right). These buttons must relate to the JSON data on each click (selection).

Let us explain the functionality of using *Index* and *Page* with an example. Assume the Option menu is on the second Page, showing Index values 5–9. The top Option button of the *layout* is 0 and it will never change. That numeric value is provided to the script, so we must find a way to indicate that this layout button 0 relates to JSON Index 5. That is where the usefulness of paging comes in. Since we are on the second Page, we notice that each Page counts as a multiple of the number of buttons in the menu, so we can deduce the Page's value correctly. Indexes 0, 5, and 10 would now be (Button 0, Page 0), (Button 0, Page 1), and (Button 0, Page 2). Writing scripts to use the paging approach ensures that buttons can maintain their hardcoded values in the Unity Editor. It also allows us to write our functions solely based on which button is pressed with its easily referenced Page number, using simple math.

3. Groups

Groups is another field that allows for extendibility. We chose ten similar networking commands as a well-rounded set, but we can easily create commands for a different set of activities like forensics, processes, or even local commands. That type of extension is possible because the implementation for the Options and Services concepts remains the same. Adding new sets of commands would simply consist of appending new information to the JSON. The Service menu has different groups as well. Grouping of options is not required, as a command's various options are only accessed through paging.

G. BACKSPACE AND CONTEXT SWITCHING

Pressing a new command changes the application's state and requires the system to feature that command's options, and thus the need to switch the system state to a new context. Initially, we only appended *text* to an array upon each click of a menu button. While that provides a valid string, it inadvertently did not provide a mechanism for going back to a prior layout, a prior state, a prior context, upon backspacing. For example, when "ping -4" is selected, all options available to the GUI belong to ping. If the user decides to add to the command, and use piping (e.g., "| arp -a"), then the final command will be "ping -4 | arp -a" and all options will be specific to arp. Now, because backspace only deletes the group of characters associated with the last button click, if we were to backspace 3 times our command would state "ping -4" again. Yet because we did not remodify the application state, all options would still belong to arp. That would be incorrect from the standpoint of the application state (i.e., options should be specific to "ping").

Therefore, we must add and delete words, **and** add and delete entire contexts. The extra class StatusNode contains fields for Groups, Indexes, and Pages; that node acts as a global variable for any change done with the buttons. New nodes are created and stored in an array when a user selects additional Menu buttons, and 'Back' deletes the last element in the array, then consults the element that now just became the new 'end' and replaces the context as appropriate.

H. TOOLTIPS AND TOOLTIP TRIGGERS

The Tooltip system functionality displays a header and content when a user's cursor hovers over a small yellow triangle positioned in the left upper corner of the menu buttons; the only exceptions are buttons with arrows, and the 'Run', 'Back', and 'End' buttons. The author provides credit for the programming of the scripts for the Tooltip, Tooltip System, and Tooltip Triggers to Game Dev Guide, a highly valuable YouTube programming series that explains everything needed to make the tooltips more flexible and give them a professional appearance, from demonstrating straightforward actions like canvas creation and screen-size scaling, to more sophisticated activities like dynamic

anchoring and event handling (Game Dev Guide, 2020). Examples of the dynamic ability of the tooltips is shown in Figure 33.

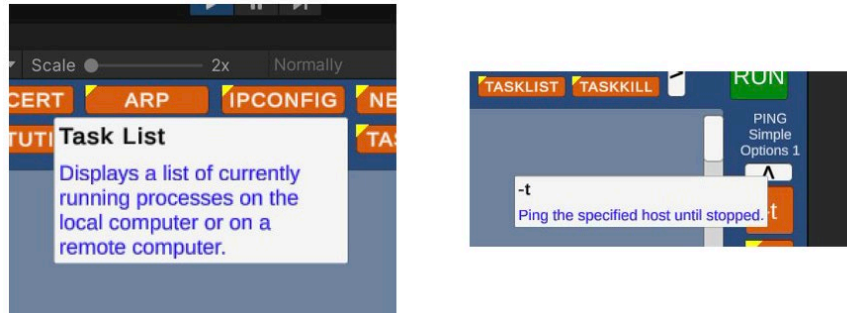


Figure 33. Dynamic Tooltips extending downwards (left) and to center (right)

Data needed for the tooltips are again imported automatically from JSON and placed on a dynamic grey canvas. The bold black header is the name of the button, and the blue content is that item's description. The “dynamic” feature positions the tooltip more to the right if the cursor was hovering over a button far to the left, and closer to the center if the cursor was hovering over one of the top command buttons, so that nothing is cut off by the edge of the window created to accommodate the GUISE GUI. As the text is written the tooltip width expands up to a certain predetermined length, then begins word wrapping to accommodate the longer content. While the exact details of the implementation are beyond the scope of this study, it is worth mentioning that we developed this functionality by following the instructions provided in the 10-minute YouTube video, and the final product resulted in a fully functioning Tooltip Trigger script that could be attached to each tooltip button.

I. OUTPUT TO FILE / INPUT TO SCREEN

A problem when reading the result of the ExecuteCommand method is that the result comes back as some abstruse string. Copying and pasting the results into the scroll menu results in a disarranged output, with characters such as tabs and spaces at improper widths. One workaround was to save the output as a text file (called “output.txt”), which

does properly format the output. The file can then be read as a string and posted to the scroll area, with proper looking text results.

J. CHAPTER SUMMARY

This chapter provided details about the software solutions that were used to develop the GUISE GUI and application. We provided specifics of the naming conventions, discussed how JSON data acts as a retainer for information, explained how Unity C# scripts allow for interaction with the GUI buttons, and showed the use of dynamic tooltips.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND RECOMMENDATIONS

A. PRINCIPAL THESIS CONCLUSIONS

The GUI for Shell Entry (GUISE) application represents a novel way to interact with the shell. It incorporates an interface with the most common commands for networking, especially suited to support the educational needs of novice users and management needs of professional users. The menu buttons that circumnavigate the console-like center screen provide access to guided commands, as do their easy-to-use tooltips, which in turn, help the user narrow down the available choices and extend the commands to be more specific or complex.

B. ADDRESSING RESEARCH QUESTIONS

This thesis has set out to achieve five design goals detailed in the chapter dedicated to the design of the GUISE application and its GUI.

1. Research Question 1: Interface for enhanced situational awareness

“What type of graphical user interface and user interactions can provide enhanced comprehension of OS commands and their capabilities to novice users?”

We addressed this research question by defining a set of design goals that we followed. We list and comment on each design goal individually.

- Design Goal 1: The user interface will provide fast and efficient probing of the most effective commands on the same interface.

A design goal was to prototype a user interface to provide fast and efficient probing of the commonly used shell commands. The ten most popular networking commands are chosen based a on careful selection from several reputable websites and Microsoft documentation. They are placed dominantly at the top of the application as a starting set of commands offered to the users. Alternate “Command Groups” are placed on the second page, with the possibility of adding new groups of commands. That feature made the application extendible. With this prototype as the backdrop, their conspicuously

placed tooltips allowed quick reviewing of each command so choices can be made efficiently and instantly.

- Design Goal 2: The interface will employ the types of user interactions that are simple, easy to use, and familiar to users with no technical expertise.

GUISE is a graphical user interface, and it does not rely solely on text; that opens the application to an audience that is averse to using the command line. The most popular commands are already shown at the top, and the fact that command descriptions can easily be read from the tooltips alleviates the memorization to use commands. Users have easy access to commands and their Service Options by selecting (clicking) the buttons available via the GUI; if ever in doubt, she can check the tooltips to get necessary information.

- *Design Goal 3:* The user interface will provide a clear situational awareness at any point of user interaction: a user will know the options offered via the interface, the way to abandon the current action, and the way to execute a desired course of action.

The commands are shown at the GUISE prompt at the top of the scroll area in real time as the user types, and the action buttons for ‘Run’, ‘End’, and ‘Back’ are clearly shown in unique colors. All Service options available via the menu are context sensitive. The user selection of the ‘Back’ button is followed by an appropriate change of the command context, resulting in correct Command or Service options being available for user selection.

2. Research Question 2: Application that is extendible

“Could the underlying system architecture and user interface be designed to support a variety of classes of OS commands and services beyond the network analysis?”

- Design Goal 4: While the interface will be focused on networking commands, its design will support easy extensions to include other groups

of commands like forensic commands, file and directory navigation, and pipes and filters.

GUISE system architecture and interface design fully support a diverse set of commands via its construct of *Groups*. The initial set shows ten networking commands, and a user can reach another group of commands with one simple click. A comprehensive set of complete commands were left out of GUISE but can be constructed. For example, it is easy to add a “Reconnaissance Group” with the most-used commands related to network discovery. GUISE does include the `tracert`, `ping`, and `route` commands to provide some level of reconnaissance, but a fully usable set would need to include `telnet`, `ftp`, `nmap`, and a download of external CLI exploitation tools like Metasploit. Again, this prototype focused on a user’s personal network and building the structure of the system. Nonetheless, Metasploit is another prime example of how a user’s lack of memorized commands and phrases can be a hinderance to popularity, and the authors look forward to bringing GUISE’s demonstrative interface to text-based exploitation and reconnaissance tools.

Our most pertinent goal was in providing general networking education through using our program continually, so that a user could gradually retain fundamental networking concepts and learn how to use the command line more proficiently. Even while creating this prototype, the author—a second year master’s student in cybersecurity—began learning many concepts and commands that had up to then been unfamiliar, despite several years of computer science study. Our team learned that once a command is clicked, there is a strong inclination to also click other buttons along the side of the menus to “see what they do.” I might have known of the `whoami` command, which shows my computer name. But with `/priv`, `/user`, `/use`, and `/logonid` buttons off to the side, those would also likely be clicked while I explore my personal network. Even though those specific items might not have been desired during this round of requests, seeing their operation at least once lets me know those features are available and contributes to heightened knowledge of commands in general.

C. LIMITATIONS

GUISE users experience a short delay when running a command selected via the GUI. The script needs to go through several lines of GUI code and must open and close a text file to present the result (i.e., system feedback for the selected command). This results in a time delay. For example, while pinging an IP address shows results instantly through the usual command prompt, it takes several seconds before results are shown through GUISE. However, for the people it was designed for, this would not be a problem for the ‘relatively’ simple lines of code they will be creating.

The execution of commands run through GUISE can take a long time to have results returned, via either processing a command that returns long text or if there is an infinite loop. An infinite loop does not necessarily indicate a problem. For instance, the command “*ping -t www.google.com*” will ping Google indefinitely. An action/capability that is an absolute requirement is one that facilitates the stopping of such a running program. This is done on the Windows command line by pressing Control-C, however this functionality was not implemented in GUISE. With further development, that capability can easily be added as an additional Service.

There are limitations to how we can add options. GUISE has a hardcoded space after each button click, so for an unimplemented option that is irregular, we must type the exact wording. An example of this is the `dir` command. For its attribute option, instead of using “-a” as is normal, it uses a backslash, “/A” as *well* as one of several extended attributes. There is a colon (“:”) between the attribute and its extension, and it cannot have any spacing in between. In order to include these options *and* their extensions, we had to write each as an individual option: /A:H for hidden files, /A:S for system files, etc. Additionally, we could not include the ‘-’ (which means ‘not’) in front of the option for the same reason, so negative options must also be written out separately.

Although many appropriate commands are highly important for networking and administration, several commands that may qualify as such were not included.

- **Deeper net, whoami, and other long-option Commands** are too complicated to fully provide all their options. Although basic options are

scripted to allow users to gain familiarity with simple usage, a complete set is better suited for future evolutions of this application.

- **Whois, TShark, and other commands that must first be downloaded** have a setup that is too intricate for our goals. A more complete installation can be included as a package in the future.

Our software concerns *personal* cyber-connectivity, focusing on verifying and exploring network information of a single host. That work did not include modifying any settings via commands in this iteration of GUISE. More useful commands than the ones we included would simply require appending to the JSON a few levels deeper than we went, or they may require the addition of another field or method in the C# script. While these actions are achievable now, we kept the baseline of all commands as similar as possible; each of our Menu items chains a simple or parameterized option with a possible final target string. Anything outside of that is beyond the scope of the thesis.

D. RECOMMENDATIONS FOR FUTURE USE

The opportunity statement of GUISE can be shown below.

1. GUISE as a Security Management Tool

With GUISE put in the hands of a novice user, we hope the easy-to-read buttons, quick referencing of commands through tooltips and error correction, as well as its extensibility, will make it a valuable product. Beginners and non-technical workers in charge of cyber-vulnerable equipment should find very good use for this.

With GUISE operated by amateur computer users, we hope this could be their first introduction to networking as they slowly but gradually memorize commands and become power users. Indeed, they would see high value for the custom buttons that could be created for their most common CLI commands.

With GUISE operated by a professional cyber administrator, reconnaissance can be scripted and performed easily and quickly through the custom buttons. Company scripts can be created by the administrator and passed on to workers so they can perform

complex cyber validations with only a few buttons and zero command memorization. This activity could have been beneficial in the examples of the Colonial Pipeline and Florida Water hacks mentioned in the Background chapter; a few select buttons could have been crafted for non-IT users to occasionally monitor remote connections and kill or suspend suspicious processes.

2. GUISE as an Educational Tool

The GUISE application can serve as a pedagogical tool to learn about the networks and swiftly use relevant commands. The convenient display of the most popular commands and their options, shown in plain view, is the main way users can increase their knowledge of the system, followed by reading the normally hidden tooltips. The unassuming and colorful layout of GUISE gives it a welcoming appearance, and our hope is that users of all age levels will not be apprehensive to use it, from the 10-year-old turning on his first PC to the office worker who notices her screen start to flicker.

Tutorials can be created that ask random questions to build networking maturity, such as, “Type the commands to determine if your computer can connect to www.nps.edu.” The software could guide a user to hints and, most importantly, give answers immediately. GUISE can also be used to support other game-like experiences. With objective goals presented to them as a challenge, users can have fun by repeatedly clicking and learning. Amusing games can also be crafted where a user must use our software to find the problem with a faulty network or to hunt down an exploiter on the system.

3. Microsoft Windows Integration

We suggest including this application in each new Microsoft operating system, starting with the home version, as an add-on or free download. This backward capability allows the years of collected troubleshooting information to remain relevant to our software as well since GUISE runs through executing the same Microsoft command prompt command, while ingraining all visible buttons in memory with each usage.

4. Open-Source Project

There is much room to customize and create new command groups and make this an open-source project. That would include continuous expansion of the code for more features, covering additional operating systems, providing more examples when needed, and fixing bugs.

E. FUTURE WORK

We recommend the following extensions of the work presented in this thesis.

1. Usability Study

While no user study was performed using GUISE, an investigation can be constructed to determine the efficiency of finding the correct commands compared to traditional command prompt use, and the effectiveness of typing in more complex (with more sequential and varying options) and chained commands, using pipes and redirects. The same study would also collect subjective responses on user satisfaction with different design solutions available in the GUI, the functionality of the application, and potential improvements of the GUISE interface.

2. User Study Focused on Learning Effectiveness

Studies on the learning effectiveness and recall after not using the system for an extended time could also be executed. Ideally, we want a user familiar with using GUISE at home to maintain the knowledge learned from home-use, even if using the command prompt on a computer at work. The same skill level should ideally be maintained for some time, even if one is not using GUISE or the command line daily. Additionally, users should not have difficulty using GUISE again after an extended period of non-use.

3. Implementation on Other Operating Systems

Because the execution can be specific to an operating system, we ran our software only on the Microsoft Windows operating system. The JSON data has a field that dictates the current operating system, and code can easily be modified to change based on the

type of OS the user is running. That could be a worthwhile endeavor with the increasing number of Mac OS X home users and the popularity of UNIX-based servers.

LIST OF REFERENCES

- Avantgarde Software. (2019, December 17). [Unity] *Coding guidelines & basic best practices*. <https://avangarde-software.com/unity-coding-guidelines-basic-best-practices/>
- BECS Software Solutions. (2020, October 30). *Malwarebytes free vs premium | is malwarebytes free good enough?* <https://www.becs.co.uk/malwarebytes-free-vs-premium/>
- Bing, C. J. M. (2021, June 8). *U.S. seizes \$2.3 mln in bitcoin paid to Colonial Pipeline hackers*. Reuters. <https://www.reuters.com/business/energy/us-announce-recovery-millions-colonial-pipeline-ransomware-attack-2021-06-07/>
- Campion, N. (2020, October 30). Pros and cons of a command-line interface. *The Iron.Io Blog*. <https://blog.iron.io/pros-and-cons-of-a-command-line-interface/>
- Chandel, R. (2020, February 9). *Beginners guide to TShark (Part 1)*. Hacking Articles. <https://www.hackingarticles.in/beginners-guide-to-tshark-part-1/>
- Computer Hope. (2021, August 16). *Command line vs. GUI*. <https://www.computerhope.com/issues/ch000619.htm>
- Cruz Sendy, S. (2015, January 12). *Create computer network with Cisco Packet Tracer Part I* [Video]. [www.youtube.com, https://www.youtube.com/watch?v=q-UUbPk6fYo](https://www.youtube.com/watch?v=q-UUbPk6fYo)
- Dictionary.com. (2022). Guise. In *Dictionary.com*. <https://www.dictionary.com/browse/guise>
- DVI Aviation. (2022). *Airplane, aircraft cockpit design experts*. <http://www.dviaviation.com/aircraft-cockpit-design.html>
- Dynatrace. (2022). *Dynatrace comparison*. <https://www.dynatrace.com/platform/comparison>
- Fisher, T. (2022, February 4). *Command Prompt: What is it and how to use it?* Lifewire. <https://www.lifewire.com/command-prompt-2625840>
- Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6), 381–391. <https://doi.org/10.1037/h0055392>
- Game Dev Guide. (2020, November 4). *Designing a responsive tooltip system in Unity* [Video]. <https://www.youtube.com/watch?v=HXFoUGw7eKk>

- Garreau, J. (1994, March 9). Office Minefield. *The Washington Post*.
<https://www.washingtonpost.com/archive/lifestyle/1994/03/09/office-minefield/3b74132a-5f0a-455f-a04e-6171d023149b/>
- Gibson, J. J., & Carmichael, L. (1966). *The senses considered as perceptual systems*. Boston: Houghton Mifflin.
- Hoffman, C. (2017, August 28). *How PowerShell differs from the Windows command prompt*. How-to-Geek. <https://www.howtogeek.com/163127/how-powershell-differs-from-the-windows-command-prompt/>
- Kellerer, J., Nikolaus Rabi, A., Neujahr, H., Möller, C., & Sandl, P. (2011). Panoramic displays: The next generation of fighter aircraft cockpits. *SAE International Journal of Aerospace*, 751–761. <https://doi.org/10.4271/2011-01-2526>
- Kelly, S., & Resnick-ault, J. (2021, June 8). *One password allowed hackers to disrupt Colonial Pipeline, CEO tells senators*. Reuters.
<https://www.reuters.com/business/colonial-pipeline-ceo-tells-senate-cyber-defenses-were-compromised-ahead-hack-2021-06-08/>
- Kothari, J. (2021). *Comprehensive guide on TShark*. GNite Technologies.
<https://dl.packetstormsecurity.net/papers/general/tshark.pdf>
- Lee, C. (2016, January 26). Apple updates Airport Utility for IOS with new features and improvements. *IDownloadBlog.Com*.
<https://www.idownloadblog.com/2013/02/07/apple-updates-airport-app/>
- Malwarebytes. (2022). Malwarebytes cybersecurity for home and business | Anti-malware & antivirus. <https://www.malwarebytes.com/>
- Marquardt, A., Levenson, E., & Tal, A. (2021, February 10). *Florida water treatment facility hack used a dormant remote access software, sheriff says*. CNN.
<https://www.cnn.com/2021/02/10/us/florida-water-poison-cyber/index.html>
- Microsoft | Docs. (2022, January 4). *Windows commands*. <https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/windows-commands>
- Paessler. (2022). *Take control of your bandwidth | Network traffic monitoring with PRTG*. https://www.paessler.com/network_traffic_analyzer
- Posey, B. (2017, August 17). *11 Networking commands every windows admin should use*. TechGenix. <https://techgenix.com/top-11-networking-commands/>
- Rogers, E. M. (2003). *Diffusion of Innovations*. Simon and Schuster.
- Russinovich, M. (2022, February 16). *Sysinternals - Windows Sysinternals*. Microsoft | Docs. <https://docs.microsoft.com/en-us/sysinternals/>

- Skillshare. (2022). Explore thousands of hands-on creative classes.
<https://www.skillshare.com>
- SolarWinds. (2022). *Network performance monitor*.
<https://www.solarwinds.com/network-performance-monitor>
- Swade, T. (2020). *Networking commands | Top 9 commands of networking*. EDUCBA.
<https://www.educba.com/networking-commands>
- Tognazzini, B. (1992). *TOG on interface*. Pearson Education.
- Udemy. (2022). *Udemy*. <https://www.udemy.com>
- Unity Technologies. (2022a). Unity real-time development platform | 3D, 2D VR & AR engine. <https://unity.com>
- Unity Technologies. (2022b, March 12). *Unity—manual: TextMeshPro*.
<https://docs.unity3d.com/Manual/com.unity.textmeshpro.html>
- Unwin, A., & Hofmann, H. (1999). GUI and command-line—Conflict or synergy?
Computing Science and Statistics, Proceedings of the 31st Symposium on the Interface, 246–253.
<http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=FE38A63D4210377E3DE084588D3F4562?doi=10.1.1.31.1928&rep=rep1&type=pdf>
- Vijayalakshmi, M., Dasai, P., & Raikar, M. (2016). Packet Tracer Simulation Tool as pedagogy to enhance learning of computer network concepts. *2016 IEEE 4th International Conference on MOOCs, Innovation and Technology in Education*, 71–76. <https://doi.org/10.1109/MITE.2016.024>
- Wilke, C. (2021, June 8). *Colonial Pipeline paid \$5 million ransom one day after cyberattack, CEO tells senate*. CNBC.
<https://www.cnbc.com/2021/06/08/colonial-pipeline-ceo-testifies-on-first-hours-of-ransomware-attack.html>
- Wireshark. (n.d.). *About Wireshark*. <https://www.wireshark.org>
- World War Wings. (2022). *100 Years of fighter plane evolution is truly eye opening*.
<https://worldwarwings.com/the-evolution-of-fighter-planes-in-the-last-century-is-beyond-stunning>
- Wouk, K. (2021, December 3). *How to scan your local network with terminal on MacOS*. Maketecheasier.
<https://www.maketecheasier.com/scan-local-network-with-terminal-macos/>

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California