ENHANCING ADVANCED CYBERSECURITY EDUCATION THROUGH INCLUSIVE, ENGAGING PEDAGOGY

Presented by:
Harini Ramaprasad
University of North Carolina at Charlotte

2023 CAE in Cybersecurity Symposium
June 2023

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

CYBERSECURITY EDUCATION IS IMPORTANT

- Big gap in cybersecurity workforce demand & supply
  - ~3.14 million professionals needed to fill gap
- Lack of diversity
  - Only ~4% of cybersecurity workers in US identify as Hispanic, 9% as Black, and 24% as women
- Diversity and inclusion are not just feel-good initiatives
  - Essential for protecting critical infrastructure
  - Leads to creative, varied-perspective solutions to challenging problems
- **Our goal**: *Teach advanced cybersecurity topics in an inclusive, engaging manner*

Courtesy: forbes.com

Minorities and the Cybersecurity Skills Gap, *Forbes Technology Council,* Sept 2022

E-SHIIELD: Enhancing Security education in Hybrid mobile and IoT firmware through Inclusive, Engaging Learning moDules

(NSF-funded project SaTC EDU grant # NSF-DGE1947295)













UNIVERSITY OF NORTH CAROLINA CHARLOTTE | COLLEGE OF COMPUTING AND INFORMATICS

E-SHIIELD: Enhancing Security education in Hybrid mobile and IoT firmware through Inclusive, Engaging Learning moDules

(NSF-funded project SaTC EDU grant # NSF-DGE1947295)

- *Criminal Investigations*
  - Gamified web-based framework to teach and assess Internet of Things (IoT) security skills
- *DISSAV*
  - Program visualization tool for teaching stack smashing attacks
- Suite of *guided learning* activities
  - Use Process Oriented Guided Inquiry Learning (POGIL) style
  - Start from foundational concepts and build up to stack smashing attacks and defenses

UNIVERSITY OF NORTH CAROLINA CHARLOTTE | COLLEGE OF COMPUTING AND INFORMATICS

CRIMINAL INVESTIGATIONS

GAMIFIED, SCALABLE WEB-BASED FRAMEWORK FOR TEACHING AND ASSESSING IOT SECURITY SKILLS

UNIVERSITY OF NORTH CAROLINA CHARLOTTE | COLLEGE OF COMPUTING AND INFORMATICS

CRIMINAL INVESTIGATIONS

GOALS

- Promote student learning and engagement
- Motivate students to explore advanced topics in cybersecurity
- Promote inclusivity, accessibility and broader dissemination
- Deliver IoT educational content in an engaging, inclusive way

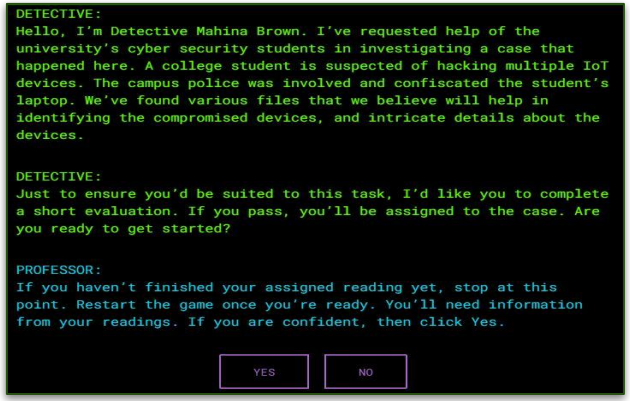
CRIMINAL INVESTIGATIONS

DESIGN

- Incorporates elements of gamification into hands-on activities
- Key focus on interactivity to promote student engagement
- Designed to be used in conjunction with other learning content (lectures, readings, tutorials, etc.)
- Accessible to students from diverse backgrounds

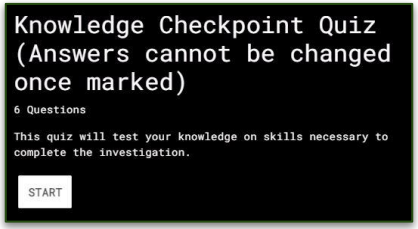
CRIMINAL INVESTIGATIONS

DESIGN



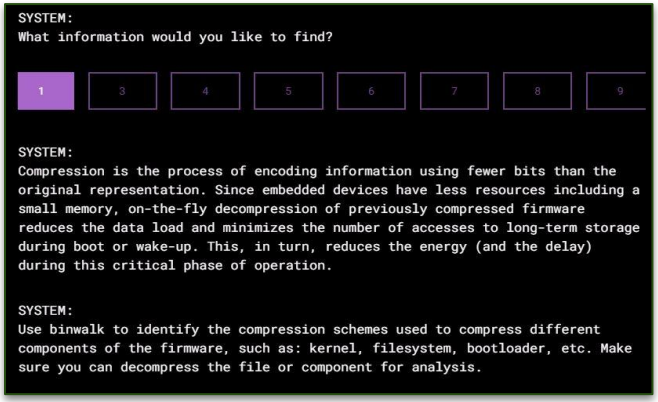
Narrative

Keeps students motivated



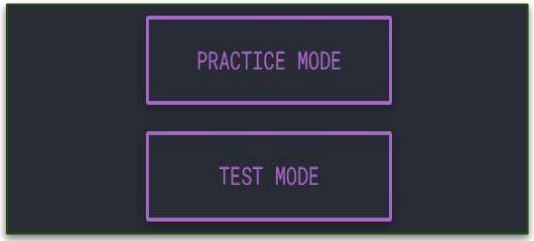
Knowledge checkpoints

Assess student preparedness



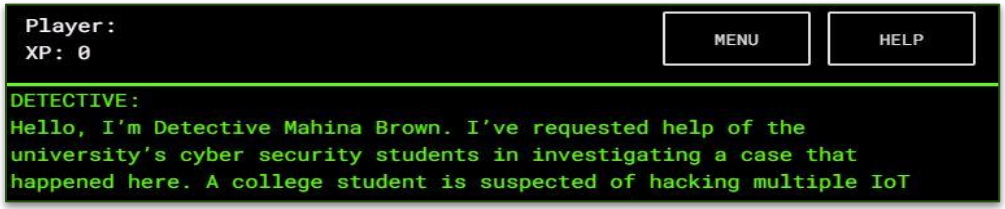
Just-in-time learning content

Reinforce key concepts while engaged in activity



Game modes

Keeps students motivated, interactive



eXperience Points

Keeps students motivated

CRIMINAL INVESTIGATIONS

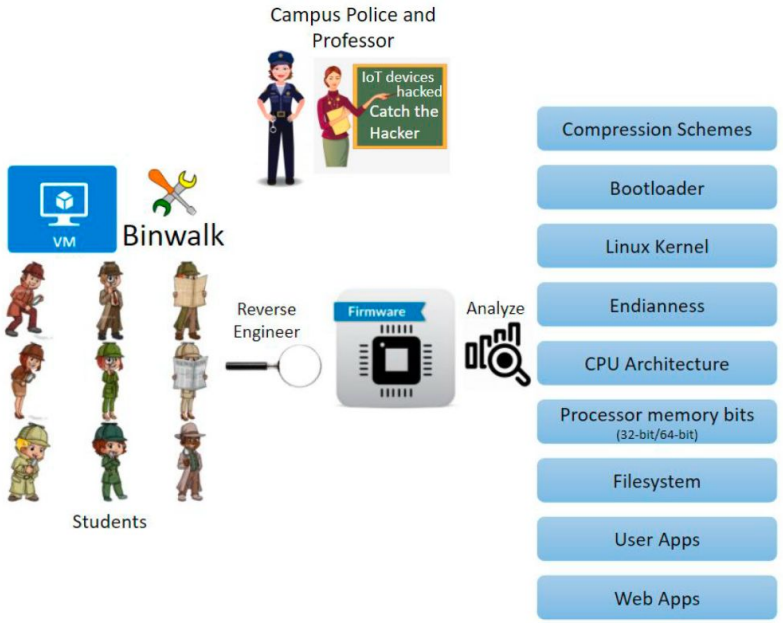
IMPLEMENTATION

- Web-based application
  - React for UI and front-end
  - Python Flask library for backend, MongoDB for backend database
- Deployed on UNC Charlotte server that runs Ubuntu 18.04 LTS
- Accessible to students on campus or through VPN
- All required tools and files provided within pre-built Virtual Machine (VM) image

CRIMINAL INVESTIGATIONS

PROTOTYPE ACTIVITY: REVERSE ENGINEERING AND ANALYZING IOT FIRMWARE

- Reverse engineer an IoT firmware image using binwalk
- Identify firmware components: compression schemes, kernel, bootloader, filesystem, user apps, web apps, CPU endianness, architecture, and processor type
- Compelling narrative placing students as assistant investigating campus IoT hacking incident
- 9 non-sequential activity tasks, each accompanied by short summary and relevant security information
- VM w/ binwalk and dependencies pre-installed and accessible from terminal



CRIMINAL INVESTIGATIONS

PILOT STUDY: SETUP

Survey to gauge student perception of guided learning activities

- 16 Likert scale questions - feedback on UI (2), learning (6) engagement (8)
- 5 free response questions - issues, bugs, strengths, improvements, additional feedback

Deployment, Fall 2021

- 3 sections of junior level undergraduate intro to OS & Networks course
- 1 section of senior level / early grad game design & development course [extra credit]
- 36 students completed survey and consented to have responses collected & analyzed

CRIMINAL INVESTIGATIONS

PILOT STUDY: CONCLUSIONS

- User interface: positive
- Student learning
  - Mostly positive responses on relevance to interests & targeted concepts
  - Some neutral / negative reactions on learning content / clarity of instructions
- Student engagement
  - Majority of positive responses to activity style, narrative, XP, and level of challenge
  - Some neutral / negative reactions
- Additional feedback
  - Installation issues, need for better accompanying learning content

CRIMINAL INVESTIGATIONS

ONGOING AND FUTURE WORK

- New, refactored framework using MERN stack that will allow for quick creation and deployment of new modules / activities
- Improved support for just-in-time learning content and hints
- Activities with increasing levels of complexity and progression requirements
- Ability to earn incentives and unlock challenge levels based on earned XP
- Increased randomization and adaptivity using concepts of Artificial Intelligence

CRIMINAL INVESTIGATIONS

SCHOLARSHIP

John Grady Hall, Abhinav Mohanty, Pooja Murarisetty, Ngoc Diep Nguyen, Julio César Bahamón, Harini Ramaprasad and Meera Sridhar. Criminal Investigations: An Interactive Experience to Improve Student Engagement and Achievement in Cybersecurity courses. In Proceedings of the *53rd ACM Technical Symposium on Computer Science Education (SIGCSE),* March 2022.

Abhinav Mohanty, Pooja Murarisetty, Ngoc Diep Nguyen, Julio Bahamon, Harini and Meera Sridhar. Criminal Investigations: An Interactive Experience to Improve Student Engagement and Achievement in Cybersecurity courses. Poster presented at the *52nd ACM Technical Symposium on Computer Science Education (SIGCSE'21)*, March 2021.

SIGCSE
Special Interest Group
Computer Science Education

University of North Carolina Charlotte | College of Computing and Informatics

STACK SMASHING

STACK SMASHING

THE PROBLEM & MOTIVATION

- Stack-based buffer overflow attack
  - Buffer overflow attack: Attacker writes data to buffer that overflows buffer's capacity, overwriting adjacent memory locations
    - Common vulnerability in (legacy) C programs
  - Overwrite return address to redirect program execution
- Why is it important to teach stack smashing attacks?
  - Known to be some of the most dangerous types of vulnerabilities
  - Allows remote code execution or privilege escalation
  - Affect a wide range of IoT devices
    - IP cameras, desktop conferencing IoT gadgets, Cosori Smart Air Fryer…

STACK SMASHING

THE PROBLEM & MOTIVATION

- Challenges in teaching stack smashing attacks
  - Highly sophisticated attack
  - Abstract and complex
  - C is particularly difficult
  - Requires vast background information
    - Parameter passing in C, how parameters are stored on the stack, process memory layout, many more concepts…

DISSAV: DYNAMIC INTERACTIVE STACK SMASHING ATTACK VISUALIZATION

DISSAV

Dynamic Interactive Stack Smashing Attack Visualization

UNIVERSITY OF NORTH CAROLINA CHARLOTTE | COLLEGE OF COMPUTING AND INFORMATICS

DISSAV

OVERVIEW

- Program visualization tool for teaching stack smashing attacks
- Web-based application built with ReactJS
- DISSAV workflow (a simulated attack scenario):
  - Create a function (with a buffer overflow vulnerability)
  - Construct a payload (to pass to the vulnerable function)
  - Execute the program (Attempting the stack smashing attack)
- Accompanying active learning exercise to guide students through DISSAV

DISSAV

RESEARCH QUESTIONS

We seek to answer these questions:

- (R1) Do students find that DISSAV and the active-learning exercise improve their learning of stack smashing?
- (R2) Do students find DISSAV and the active-learning exercise to be engaging resources for learning about stack smashing?
- (R3) Do DISSAV and the active-learning exercise consistently improve students' perceived learning and engagement across all age groups and genders, including students with no prior experience on the topic?

DISSAV

FEATURES

- Interactive and engaging
  - Use of colors, fonts, icons, buttons and more to improve student engagement
  - Appeal to broader and more diverse student audience
- Ability to customize attack scenario (within limits)
  - Provides guided, incremental steps for completing attack
- *Dynamic* visualization
  - Displays current state of call stack during program execution
  - Helps visualize memory addresses and contents of stack frames (abstract concept for students)
- Highlights relevant parts of program code during execution
- Allows students to customize vulnerable functions
  - Choose from list of (dummy) attacker actions, e.g., “Start a remote shell” or “Wipe OS”

DISSAV

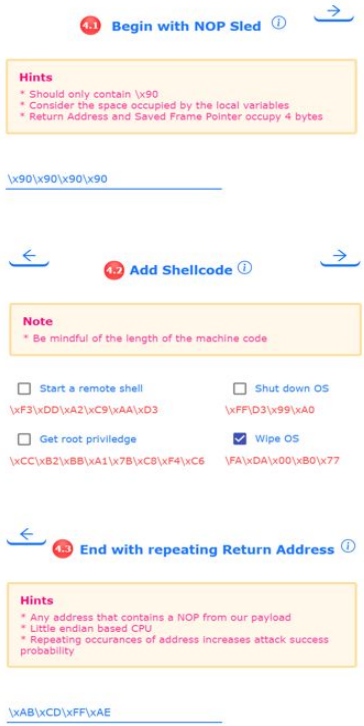
PHASE 1: CREATE A FUNCTION

- Student provides:
  - Function name
  - Local Variables (Optional)
  - Parameters (Optional)
- Additionally, student can:
  - Add call to unsafe C function
    - Currently `strcpy()`
  - Pass `argv[1]` as a parameter
  - Call another function that has been previously added to program
- Student adds function to program

DISSAV

PHASE 2: CONSTRUCT A PAYLOAD

- Separated into three parts
  - Create NOP sled
  - Add shellcode
    - Fake assembly code
    - Start remote shell, gain root privileges, etc.
  - End with repeated return address
- Separation allows student to analyze and break down each concept and work on individual pieces



DISSAV

PHASE 3: EXECUTE THE PROGRAM

- Payload is passed to function through `argv[1]`
  - During corresponding point of execution
  - Students view updated data in stack frame
    - Has return address been overwritten correctly?
    - Where in memory does new return address point to?

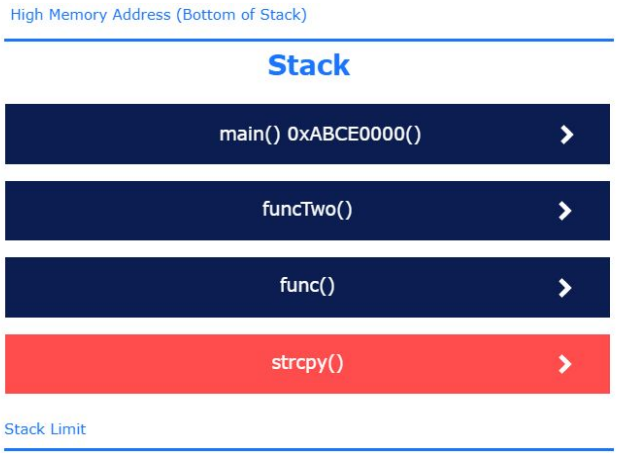
- Successful
  - Student successfully overwrites return address to point to address that contains NOP from their payload
  - Success status displays

- Unsuccessful
  - Unsuccessful status displays

DISSAV

VISUALIZATION: CALL STACK

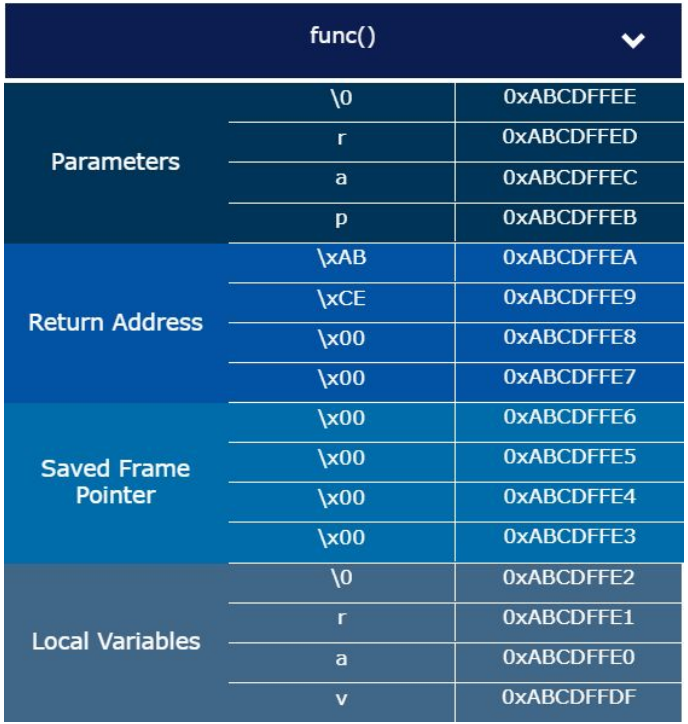
- Displays current state of call stack
  - Student clicks through to add or remove stack frame
- Dropdown to view details of function's stack frame



DISSAV

VISUALIZATION: STACK FRAME

- Displays parameters, return address, saved frame pointer, and local variables
  - Displays corresponding memory addresses
  - Provides label and color for each each section of stack frame
- Updates dynamically if student passes input to function



DISSAV

VISUALIZATION: PROGRAM CODE

- Highlights program line for each movement of stack frame
  - Highlights function name and parameters when *pushing* stack frame
  - Highlights function name when *popping* stack frame
- Tracks `argv` through program execution.
  - Dark blue font color to represent argv
  - Starts as parameter in `main` function
  - Passed to function
  - Passed to `strcpy()`

```
void funcOne(char p[]){
 char v[] = "v";
 strcpy(v, p);
}

int main(int argc, char* argv[])

 funcTwo(argv[1]);
```

DISSAV

ACCOMPANYING ACTIVE LEARNING EXERCISE

- Covers simple C programming concepts (e.g., data types) then continues to phases of DISSAV
- Provides instructions on creating vulnerable function, constructing payload, and executing function
- Encourages students to use “different strings of different lengths and number of words” before attempting to construct attack payload
- Provides guidelines for payload construction, but not exact process; students experiment by using
  - Different numbers of NOP sleds
  - Identifying and placing correct malicious return address
  - Formatting return address

DISSAV
DEPLOYMENT

Survey to gauge student perception of guided learning activities

- 14 Likert scale questions - feedback on UI (2), learning (6) engagement (6)
- 1 free response question - additional feedback
- 4 demographic questions - age, gender, prior experience with C programming, stack smashing, program visualization tools

Deployment, Fall 2021

- 2 sections of junior level undergraduate introductory cybersecurity course
  - Course introduces a broad range of security topics
  - Required course for a large number of students in our program
- Total of 104 students
  - 26 students completed survey and consented to have responses collected & analyzed

DISSAV

STUDY CONCLUSIONS

- User interface: positive
- Student learning:
  - Consistently relevant & helpful in learning targeted concepts
  - Need provide more learning resources for background concepts
  - Mostly relevant & useful, but improvement needed to tie it better to student interests & needs
- Student engagement:
  - Engaging in general, but not particularly exciting to specific groups
  - Not engrossing / immersive enough for students to feel they "lost track of time"
  - Solid resources that students would recommend to others

DISSAV
RELATED WORK

- [Sasano, BICT'15, 16]
  - Visualization tool for detecting overwritten return addresses.
  - Check (detection) if a function contains a buffer overflow vulnerability.
  - DISSAV aims to simulate an attack scenario.
- [Zhang & Yuan & Johnson & Xu & Vanamala, FIE, 20]
  - Visualization tool to teach how a buffer stores and overwrites memory.
  - Lacks an interactive call stack representation.
- Simple Machine Simulator [Schweitzer & Bolen, SIGCSE '10, 10]
  - Most closely related.
  - Visualization tool that provides dynamic visual representation of the stack during program execution.
  - Allows the user to step through a C program while viewing the stack.
  - Applies rigid rules for mapping source code to memory.
  - The instructor pre defines the SMS programs and they cannot be changed by the users during the lab.

DISSAV

SCHOLARSHIP

Erik Akeyson, Harini Ramaprasad and Meera Sridhar. DISSAV: A Dynamic, Interactive Stack-Smashing Attack Visualization Tool. *Journal of the Colloquium for Information Systems Security Education (CISSE*), (9):1, March 2022. *Best Paper Award.*

Harini Ramaprasad, Meera Sridhar, and Erik Akeyson. Interactive Program Visualization to Teach Stack Smashing: An Experience Report. *Journal of the Colloquium for Information Systems Security Education (CISSE)*, (10):1, Winter 2023.

DISSAV

Dynamic Interactive Stack Smashing Attack Visualization

UNIVERSITY OF NORTH CAROLINA CHARLOTTE | COLLEGE OF COMPUTING AND INFORMATICS

GUIDED LEARNING ACTIVITIES

POGIL®

UNIVERSITY OF NORTH CAROLINA CHARLOTTE | COLLEGE OF COMPUTING AND INFORMATICS

GUIDED LEARNING ACTIVITIES

THE PROBLEM & MOTIVATION

- Stack-based buffer overflow attack
  - Buffer overflow attack: Attacker writes data to buffer that overflows buffer's capacity, overwriting adjacent memory locations
    - Common vulnerability in (legacy) C programs
  - Overwrite return address to redirect program execution
- Why is it important to teach stack smashing attacks?
  - Known to be some of the most dangerous types of vulnerabilities
  - Allows remote code execution or privilege escalation
  - Affect a wide range of IoT devices
    - IP cameras, desktop conferencing IoT gadgets, Cosori Smart Air Fryer…

GUIDED LEARNING ACTIVITIES

THE PROBLEM & MOTIVATION

- Challenges in teaching stack smashing attacks
  - Highly sophisticated attack
  - Abstract and complex
  - C is particularly difficult
  - Requires vast background information
    - Parameter passing in C, how parameters are stored on stack, process memory layout, many more concepts…

GUIDED LEARNING ACTIVITIES

CONTRIBUTIONS

- Suite of guided learning activities
  - Warm-up resource: Strings in C
  - Activity I: Buffer Overflows in C
  - Activity II: Process memory layout
  - Activity III: Stack Smashing
  - Activity IV: Defenses
- Process Oriented Guided Inquiry Learning (POGIL)
  - Students *explore* learning models that depict relevant information, then proceed to *invent* key concepts emerging from those models, and finally *apply* the concepts they invent to solve given problem
- First to develop POGIL-style activities for advanced cybersecurity topics

GUIDED LEARNING ACTIVITIES

GOALS

We seek to answer these questions

- (R1) Do students think that the guided learning activities are well-designed and help them learn about stack smashing?
- (R2) Do students think that the guided learning activities are engaging?
- (R3) Do students across multiple age groups, genders and prior experience in areas related to stack smashing have similar perceptions about the guided learning activities?

GUIDED LEARNING ACTIVITIES

DESIGN: WARM-UP RESOURCE

Provides them with prerequisite knowledge:

- How C-style strings are created, used and stored

Example

```
char str3[] = "Hi you";
```

Memory contents, starting from the beginning of the `str3` array:

Note that the `'\0'` character has automatically been included at the end of the sequence of characters specified within double quotes.

Figure from activity that shows one way in which string can be created in C and how it is stored in memory

GUIDED LEARNING ACTIVITIES

DESIGN: BUFFER OVERFLOWS IN C

Teaches students:

- How to create and run simple C programs with command-line arguments, variables, functions, and arrays
- Structure and use of C-style strings, with emphasis on the usage of unsafe string functions such as `strcpy()`

Model 1: Command-line parameters

```c
#include <stdio.h> /* needed for printf (console display)
   function */

int main (int argc, char* argv[]){
   printf("Number of strings in argv : %d\n", argc);
   printf("List of strings in argv (one per line) :\n");
   for (unsigned int i = 0; i < argc; ++i) {
       printf("%s\n", argv[i]);
   }
   return 0;
}
```

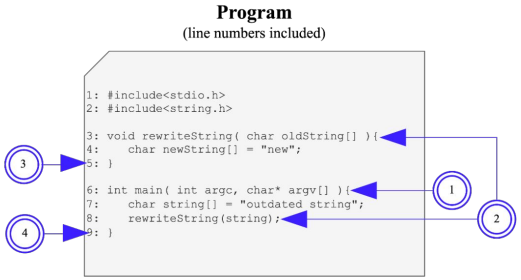
Execution command	Number of parameters passed to cmdlnpar	argc	Number of elements in argv
`./cmdlnpar`	0	1	1
`./cmdlnpar stranger things`	2		
`./cmdlnpar jon snow knows nothing`			
`./cmdlnpar ready 1 2 and 3`			6
`./cmdlnpar "this is my parameter"`			

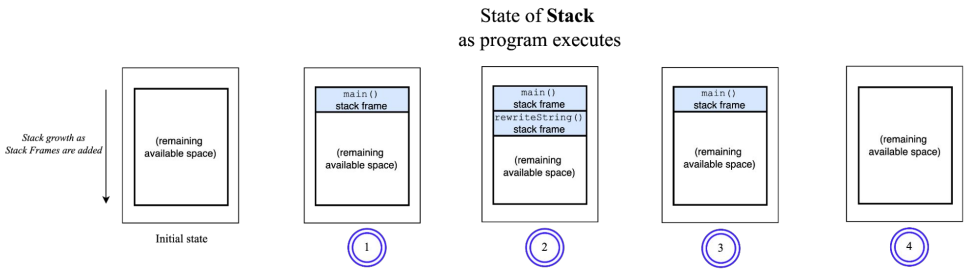
GUIDED LEARNING ACTIVITIES

DESIGN: PROCESS MEMORY LAYOUT

Teaches students:

- Purpose, relative positions, growth directions and limits of different segments within main memory of computer
- When and how stack frames are added to and removed from stack with respect to program execution
- Details of stack frame layout





Question 5 1 pts

Based on your answers to the previous two questions, describe when a function's stack frame gets added to the Stack.

- ○ When it is invoked / called
- ○ When it is returned from
- ○ Stack frames are only created for the main() function

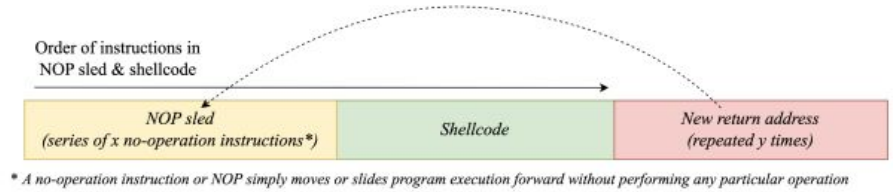
GUIDED LEARNING ACTIVITIES

DESIGN: STACK SMASHING

Teaches students:

- To recognize that unsafe user inputs
- To calculate payload size needed to overwrite return address section of given stack frame
- Purpose of NOP sled works and how to create one

Model 3:



* A no-operation instruction or NOP simply moves or slides program execution forward without performing any particular operation

Typical structure of attack payload used in practice

Question 13 2 pts

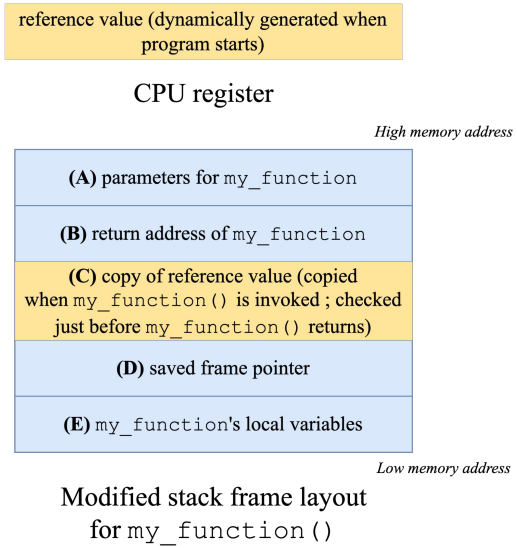
Assume that the existing return address on the stack frame is successfully overwritten with the value of the *new return address*. Will redirecting program execution as per your answer to the previous question eventually result in the execution of the shellcode as intended? Why or why not? Hint: Discuss in your group what type of instruction is at the location where your program gets redirected to and what that does.

GUIDED LEARNING ACTIVITIES

DESIGN: DEFENSES

Teaches students:

- Address Space Layout Randomization
- Non-executable stacks
- Stack canary
- Using safe C functions like `strncpy()`



Modified stack frame layout for `my_function()`

Question 16 1 pts

What purpose do you think component (C) on `my_function()`'s stack frame serves?

- ○ It can be used to prevent overflowing the local buffer
- ○ It can be used to detect if the return address has been changed
- ○ It can be used to detect malicious shellcode
- ○ It does not seem to serve any useful purpose

GUIDED LEARNING ACTIVITIES

STUDY DESIGN & DEPLOYMENT

Survey to gauge student perception of guided learning activities

- 17 Likert scale questions - student perception of length, challenge, style, outcomes, engagement and team role usage
- 1 free response question - additional feedback
- 4 demographic questions - age, gender, prior experience with C programming and stack smashing

Deployment

- 2 sections of junior level undergraduate introductory cybersecurity course, Fall 2022
  - course introduces a broad range of security topics
  - required course for a large number of students in our program
- Total of 90 students
  - 77 students completed survey and consented to have responses collected & analyzed

GUIDED LEARNING ACTIVITIES

STUDY CONCLUSIONS

- Structure and design of activities: positive responses
- Sufficiency of activities at teaching them the material: neutral reactions
- Whether the style of the activities were engaging: split
- Students younger than 25, with some prior experience with C → better perceptions of activities

GUIDED LEARNING ACTIVITIES

SCHOLARSHIP

- Harini Ramaprasad, Islam Obaidat, and Meera Sridhar. A Guided Learning Activity Suite for Teaching Stack Smashing Attacks & Defenses. In submission, 2023.

Submitted to POGIL:

- Ramaprasad, H., Sridhar, M., & Snyder, Y. (2021). Activity 1: Introduction to C. POGIL Activity Clearinghouse, 2(3). Section: Activities for Review.
- Ramaprasad, H. (2022). Process Memory Layout: Cybersecurity. POGIL Activity Clearinghouse, 3(4). Section: CS-POGIL Activity Writing Program (part of Activities for Classroom Testing).

THANK YOU!

Contact: Harini Ramaprasad (hramapra@uncc.edu)

REFERENCES

- Moog, R. S., & Spencer, J. N. (2008). Process oriented guided inquiry learning (POGIL) (Vol. 994, pp. 1-13). Washington, DC: American Chemical Society.
- Ben Allen, Minorities And The Cybersecurity Skills Gap, Forbes, 2022.
- Mohanty, A., Murarisetty, P., Nguyen, N. D., Bahamon, J. C., Ramaprasad, H., & Sridhar, M. (2021). Criminal Investigations: An Interactive Experience to Improve Student Engagement and Achievement in Cybersecurity courses. In Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (pp. 1276-1276).
- Mohanty, A., Obaidat, I., Yilmaz, F., & Sridhar, M. (2018). Control-hijacking vulnerabilities in IoT firmware: A brief survey. In The 1st International Workshop on Security and Privacy for the Internet-of-Things (IoTSec'18).
- Akeyson, E., Ramaprasad, H., & Sridhar, M. (2022, March). DISSAV: A dynamic, interactive stack-smashing attack visualization tool. In Journal of The Colloquium for Information Systems Security Education (Vol. 9, No. 1, pp. 8-8).
- Ramaprasad, H., Sridhar, M., & Akeyson, E. (2023, March). Interactive Program Visualization to Teach Stack Smashing: An Experience Report. In Journal of The Colloquium for Information Systems Security Education (Vol. 10, No. 1, pp. 8-8).