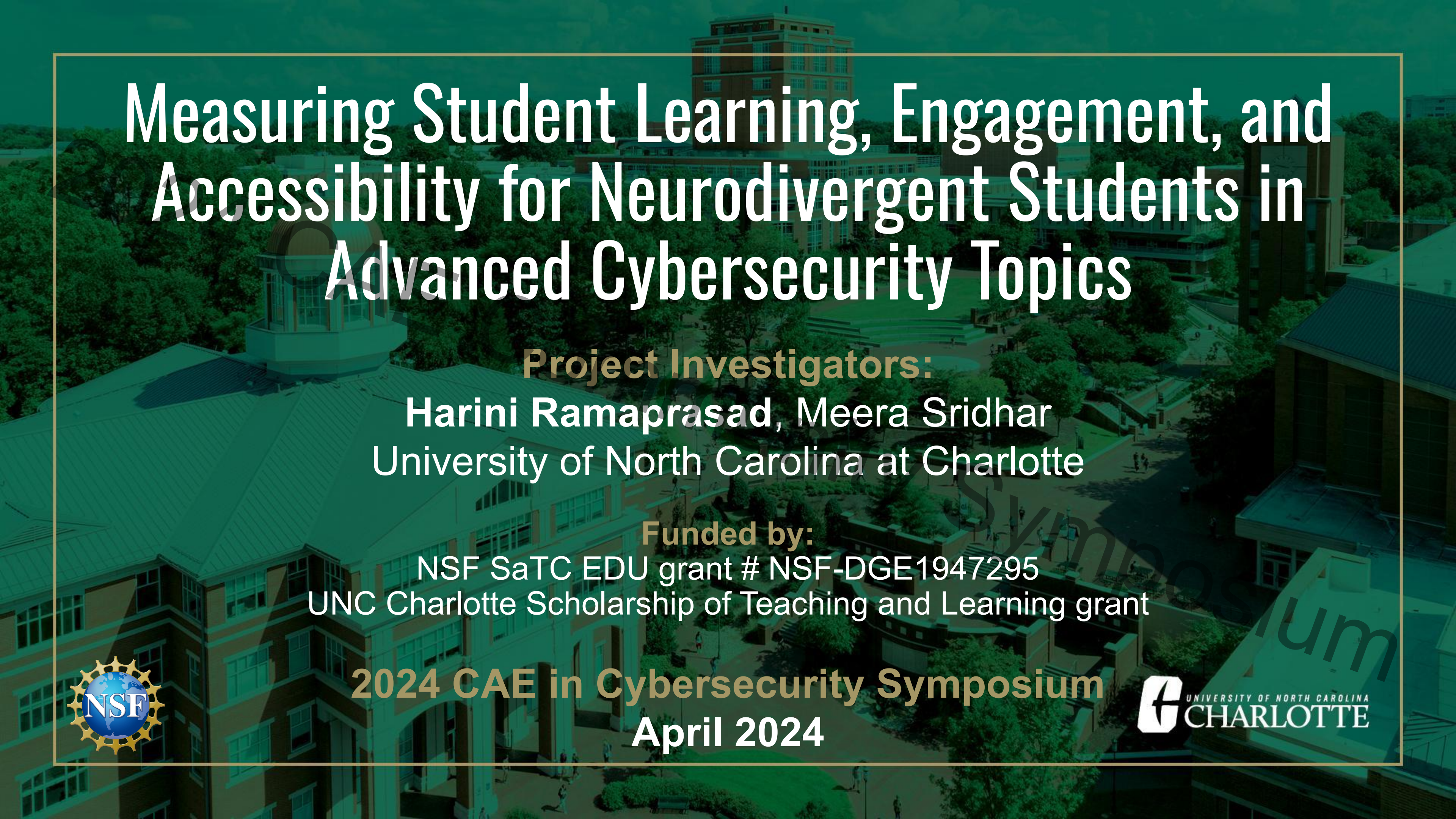


Measuring Student Learning, Engagement, and Accessibility for Neurodivergent Students in Advanced Cybersecurity Topics

Project Investigators:

Harini Ramaprasad, Meera Sridhar
University of North Carolina at Charlotte

Funded by:

NSF SaTC EDU grant # NSF-DGE1947295
UNC Charlotte Scholarship of Teaching and Learning grant

2024 CAE in Cybersecurity Symposium

April 2024

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

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

- Diversity and inclusion are not just feel-good initiatives
  - Essential for protecting critical infrastructure
  - Leads to creative, varied-perspective solutions to challenging problems

- **Overarching goal**: *Teach advanced cybersecurity topics in an inclusive, engaging manner*

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

Courtesy: forbes.com

E-SHIIELD PROJECT TEAM

UNIVERSITY OF NORTH CAROLINA CHARLOTTE | COLLEGE OF COMPUTING AND INFORMATICS

OVERVIEW

- Stack smashing - importance and challenges in teaching & learning
- Software security module
  - Guided learning activities
  - Dynamic Interactive Stack Smashing Attack Visualization (DISSAV)
- Our prior studies on software security module
  - Setup, deployment, findings
- Our current study: Effectiveness of module for neurodivergent students
  - Objectives, research questions, study participants, data collection, evaluation

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…

GUIDED LEARNING ACTIVITIES

GUIDED LEARNING ACTIVITIES

CONTRIBUTIONS

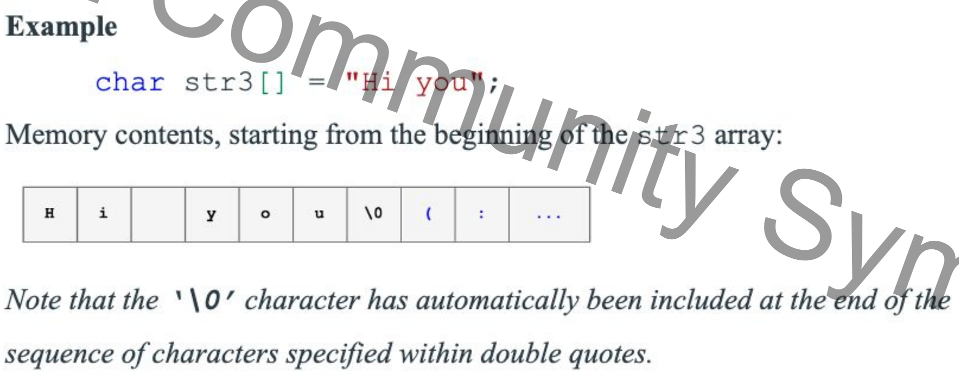
- Suite of guided learning activities
  - Warm-up resource: Strings in C
  - Activity I: Buffer Overflows in C
  - Activity II: Process memory layout
  - Activity III: Stack Smashing
  - Activity IV: Defenses
- Use Process Oriented Guided Inquiry Learning (POGIL) style
  - Students *explore* learning 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 topic such as stack smashing

GUIDED LEARNING ACTIVITIES

WARM-UP RESOURCE

Provides students 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:

H	i		y	o	u	\0	(	:	...

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

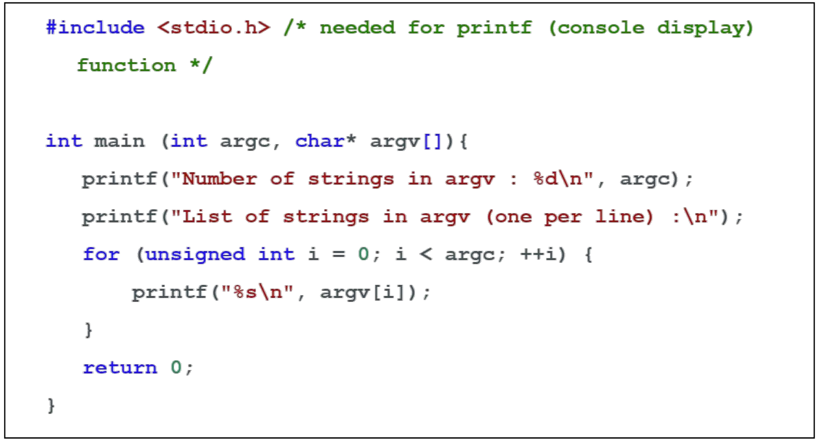
GUIDED LEARNING ACTIVITIES

ACTIVITY I: BUFFER OVERFLOWS IN C

Students learn:

- 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



```
#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"`			

UNIVERSITY OF NORTH CAROLINA CHARLOTTE | COLLEGE OF COMPUTING AND INFORMATICS

GUIDED LEARNING ACTIVITIES

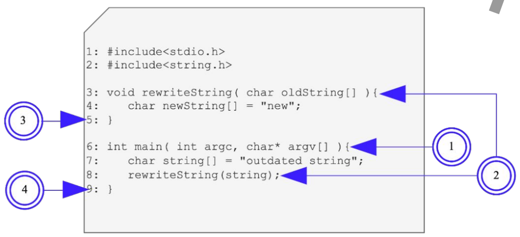
ACTIVITY II: PROCESS MEMORY LAYOUT

Students learn:

- 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

Program

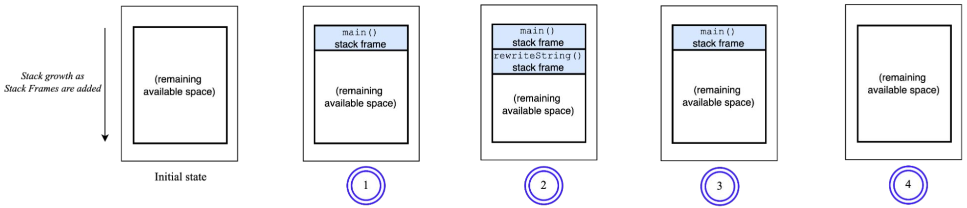
(line numbers included)



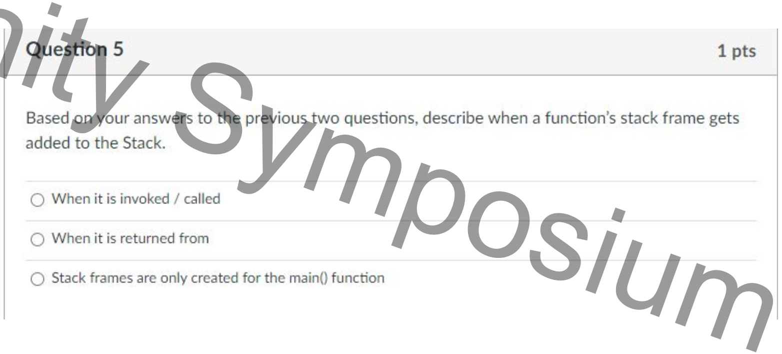
```
1: #include<stdio.h>
2: #include<string.h>

3: void rewriteString( char oldString[] ){
4:     char newString[] = "new";
5: }

6: int main( int argc, char* argv[] ){
7:     char string[] = "outdated string";
8:     rewriteString(string);
9: }
```

State of **Stack**

as program executes



Stack growth as
Stack Frames are added
(remaining available space)
Initial state
main()
stack frame
(remaining available space)
1
main()
stack frame
rewriteString()
stack frame
(remaining available space)
2
main()
stack frame
(remaining available space)
3
(remaining available space)
4



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

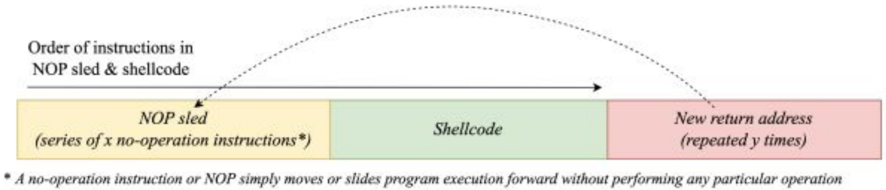
GUIDED LEARNING ACTIVITIES

ACTIVITY III: STACK SMASHING

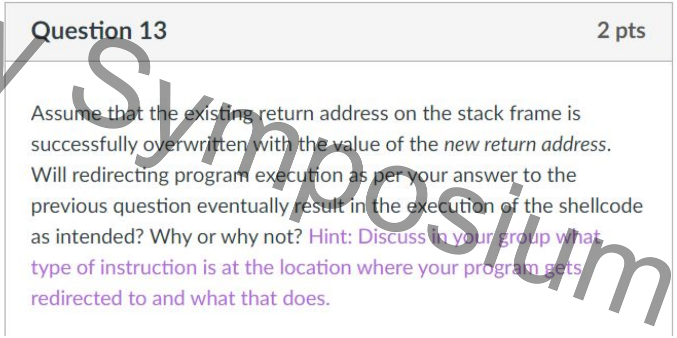
Students learn:

- 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:


Order of instructions in NOP sled & shellcode
NOP sled
(series of x no-operation instructions)*
Shellcode
New return address
(repeated y times)
* *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

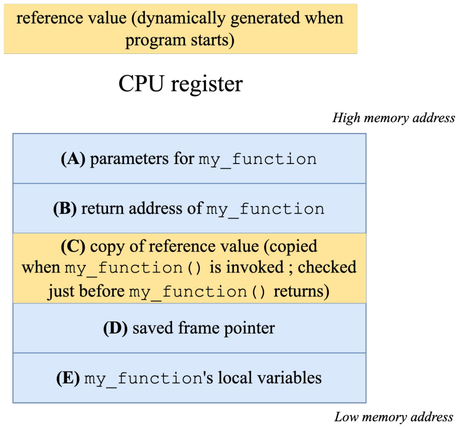
Assume that the existing return address on the stack frame is successfully overwritten with the value of the *new return address*. Will redirecting program execution as per your answer to the previous question eventually result in the execution of the shellcode as intended? Why or why not? Hint: Discuss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

GUIDED LEARNING ACTIVITIES

ACTIVITY IV: DEFENSES

Students learn:

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



reference value (dynamically generated when program starts)

CPU register

High memory address

(A) parameters for `my_function`

(B) return address of `my_function`

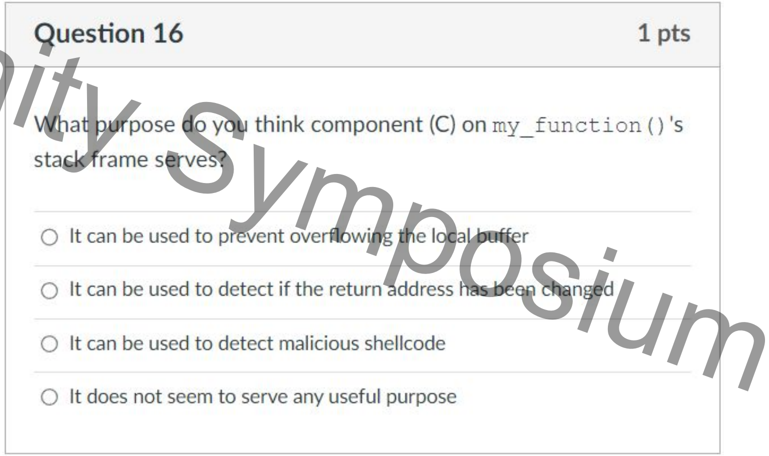
(C) copy of reference value (copied when `my_function()` is invoked ; checked just before `my_function()` returns)

(D) saved frame pointer

(E) `my_function`'s local variables

Low memory address

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

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
  - Inserted after Guided learning Activity III (Stack Smashing)

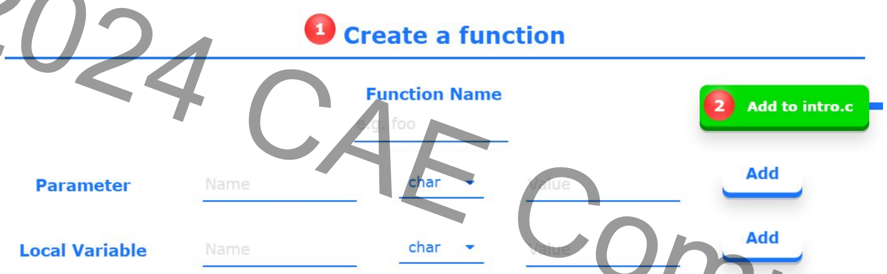
DISSAV

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

DISSAV

PHASES: CREATE A FUNCTION | CONSTRUCT PAYLOAD | EXECUTE PROGRAM & VISUALIZE STACK



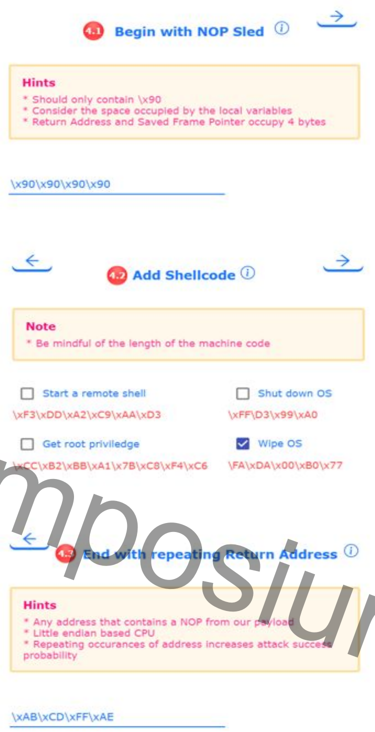
1 Create a function

Function Name: e.g. foo

2 Add to intro.c

Parameter: Name | char | Value | Add

Local Variable: Name | char | Value | Add



4.1 Begin with NOP Sled

Hints
- * Should only contain \x90
- * Consider the space occupied by the local variables
- * Return Address and Saved Frame Pointer occupy 4 bytes

\x90\x90\x90\x90

4.2 Add Shellcode

Note
- * Be mindful of the length of the machine code

□ Start a remote shell
\xF3\xDD\xA2\xC9\xAA\xD3

□ Shut down OS
\xFF\D3\x99\xA0

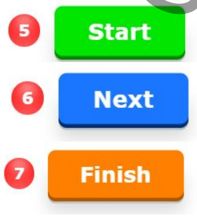
□ Get root priviledge
\xCC\xB2\xBB\xA1\x7B\xC8\xF4\xC6

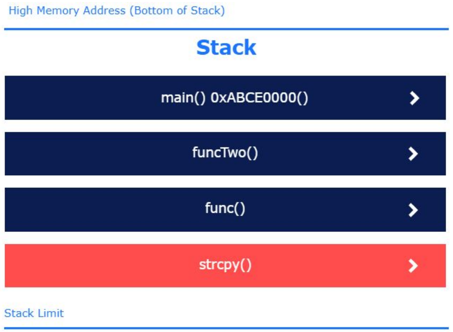
☑ Wipe OS
\FA\xDA\x00\xB0\x77

4.3 End with repeating Return Address

Hints
- * Any address that contains a NOP from our payload
- * Little endian based CPU
- * Repeating occurances of address increases attack success probability

\xAB\xCD\xFF\xAE



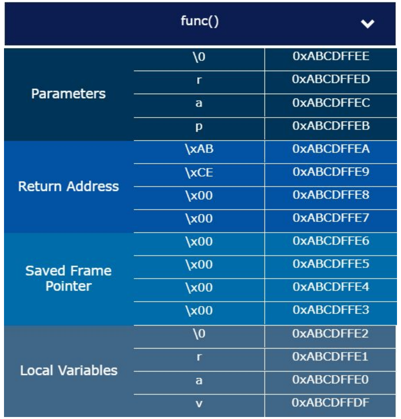


High Memory Address (Bottom of Stack)

Stack

- main() 0xABCE0000()
- funcTwo()
- func()
- strcpy()

Stack Limit



func()

Parameters	\0	0xABCDFFEE
	r	0xABCDFFED
	a	0xABCDFFEC
	p	0xABCDFFEB
Return Address	\xAB	0xABCDFFEA
	\xCE	0xABCDFFE9
	\x00	0xABCDFFE8
	\x00	0xABCDFFE7
Saved Frame Pointer	\x00	0xABCDFFE6
	\x00	0xABCDFFE5
	\x00	0xABCDFFE4
	\x00	0xABCDFFE3
Local Variables	\0	0xABCDFFE2
	r	0xABCDFFE1
	a	0xABCDFFE0
	v	0xABCDFFDF

UNIVERSITY OF NORTH CAROLINA CHARLOTTE | COLLEGE OF COMPUTING AND INFORMATICS

OUR PRIOR STUDIES

SETUP AND DEPLOYMENT

Survey to gauge student perception of DISSAV & guided learning activities

- Likert scale questions:
  - DISSAV: UI, learning, engagement
  - Guided learning activities: Length, challenge, style, outcomes, engagement and team role usage
- Free response questions for additional feedback
- Demographic questions: age, gender, prior experience with C programming, stack smashing, program visualization tools

Deployment:

- Junior level undergraduate introductory cybersecurity course across multiple semesters
  - Course introduces a broad range of security topics
  - Required course for a large number of students in our program

OUR PRIOR STUDIES

FINDINGS: DISSAV

- 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

OUR PRIOR STUDIES

FINDINGS: GUIDED LEARNING ACTIVITIES

- 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

EFFECTIVENESS FOR NEURODIVERGENT STUDENTS

EFFECTIVENESS FOR NEURODIVERGENT STUDENTS

OBJECTIVES

- Evaluate the effectiveness of our guided learning activities and DISSAV for neurodivergent students to learn about stack smashing in an effective, engaging, and accessible manner.

- Establish guidelines for the development and use of pedagogical strategies such as guide learning and program visualization to be inclusive of neurodivergent students, drawing from insights from our study.

EFFECTIVENESS FOR NEURODIVERGENT STUDENTS

RESEARCH QUESTIONS

- Do neurodivergent students think our guided learning activities help them learn about stack smashing?
- Do neurodivergent students think our guided learning activities are engaging?
- Do neurodivergent students experience challenges due to group work that is integral to guided learning activities?
- Do neurodivergent students find DISSAV useful, engaging, and accessible to learn about stack smashing?

EFFECTIVENESS FOR NEURODIVERGENT STUDENTS

STUDY PARTICIPANTS

- UNC Charlotte Cohort
  - Students enrolled in junior-level Introduction to Information Security and Privacy course, in Spring 2024 (enrollment: 50)
  - Will complete pre-test, stack smashing module, post-test as part of required coursework
  - Volunteering students will complete student experience survey (all), focus group / interview (neurodivergent)

- AccessComputing Cohort
  - NSF-funded project that focuses on increasing the participation of individuals with disabilities in computing ; ~175 team members identify as autistic, having a brain injury, learning disability, or attention deficit
  - Volunteering students will complete
    - pre-test, stack smashing module, post-test
    - student experience survey, focus group / interview

EFFECTIVENESS FOR NEURODIVERGENT STUDENTS

DATA COLLECTION

- Learning outcomes
  - Pre- and post-tests assessing student knowledge
  - Performance in guided learning activities
- Student experience survey
  - Likert scale questions about learning, engagement, accessibility
  - Demographic questions
- Focus group / interview
  - Open-ended questions on student learning, engagement, accessibility

The main objective of an attacker in a "stack smashing attack" is to overwrite the return address of a function to alter the program's control flow. True or False.

The content of the models presented in the activities was clear to me.

Using the tool helped me gain confidence in / reinforce the targeted concepts.

Were there examples, models, or activities that you found particularly helpful?

Did you encounter challenges while using the DISSAV tool? If so, how could we have designed it differently to make it easier for you?

EFFECTIVENESS FOR NEURODIVERGENT STUDENTS

EVALUATION

- Student Outcomes Data
  - Compare student performance on matched pre- and post-tests
  - Analyze distribution of student performance on guided learning style activities
  - Determine whether there are significant differences in knowledge gain and/or performance for neurodivergent versus other students in the UNC Charlotte cohort

- Student Experience Survey Data
  - Use descriptive statistics to summarize Likert-scale survey data
  - Determine whether there are significant differences in perceptions of neurodivergent and other students in the UNC Charlotte cohort
  - Use sentiment analysis for data collected from free-response survey questions.

- Student Focus Group / Interview Data
  - Qualitative data about the perceptions and experiences of neurodivergent students
  - Use combination of deductive and inductive approaches to identify themes

- Expected outcomes
  - Data analysis results
  - Guidelines for pedagogical strategies such as guided learning and program visualization to be inclusive of neurodivergent students

PRODUCTS

- 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.

- Harini Ramaprasad, Meera Sridhar, Sushma I Dangeti, Soham Pradhan. Guided Learning and Interactive Visualization for Teaching & Learning Stack Smashing Attacks & Defenses: Experiences and Evaluation. **In submission** to Frontiers in Education (FIE), 2024.

Activity approved for classroom testing via POGIL Activity Clearinghouse:

- Ramaprasad, H. (2022). Process Memory Layout: Cybersecurity. *POGIL Activity Clearinghouse*, *3*(4).

Activities submitted for classroom testing via POGIL Activity Clearinghouse:

- Activity 1: Buffer Overflows in C
- Activity 3: Stack Smashing
- Activity 4: Defenses

THANK YOU!

Contact: Harini Ramaprasad (hramapra@charlotte.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., 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).