Decompiling the Synergy:
An Empirical Study of Human–LLM Teaming in Software Reverse Engineering

Zion Leonahenahe Basque, Samuele Doria, Ananta Soneji, Wil Gibbs, Adam Doupé, Yan Shoshitaishvili, Eleonora Losiouk, Ruoyu Wang, Simone Aonzo

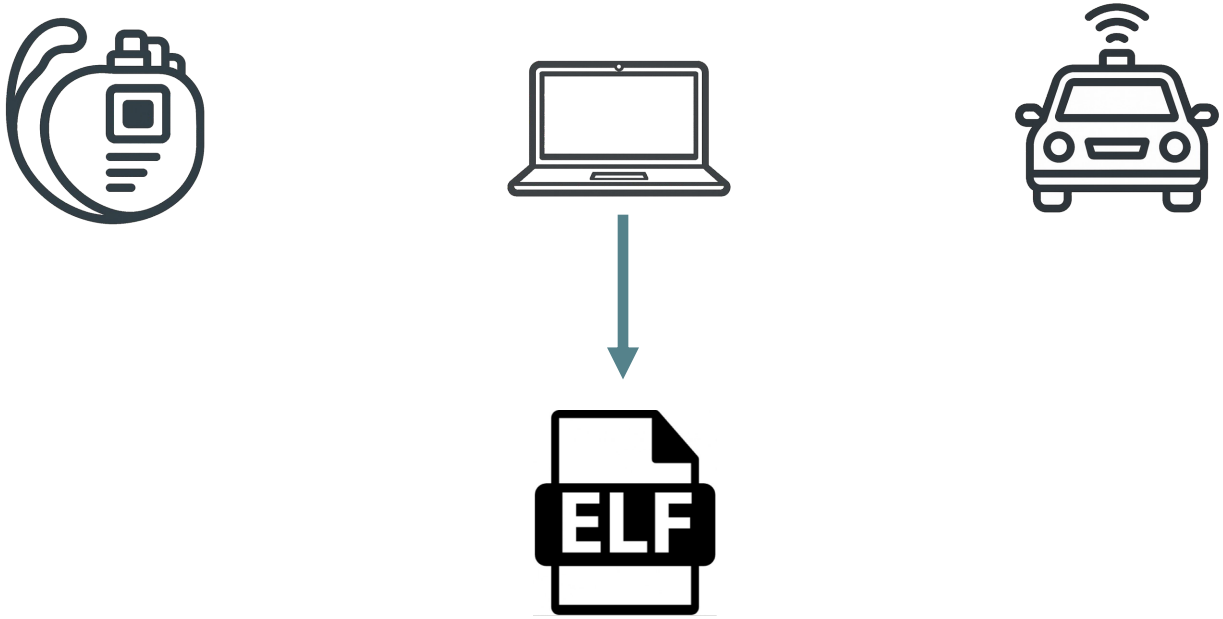
Università degli Studi di Padova

Arizona State University

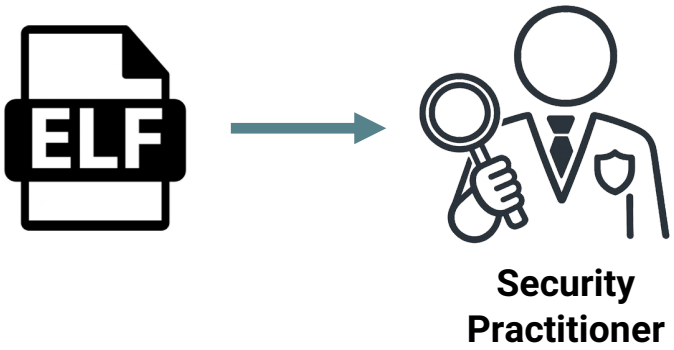
EURECOM
Sophia Antipolis

Securing Critical Software

Securing Critical Software



Securing Compiled Software



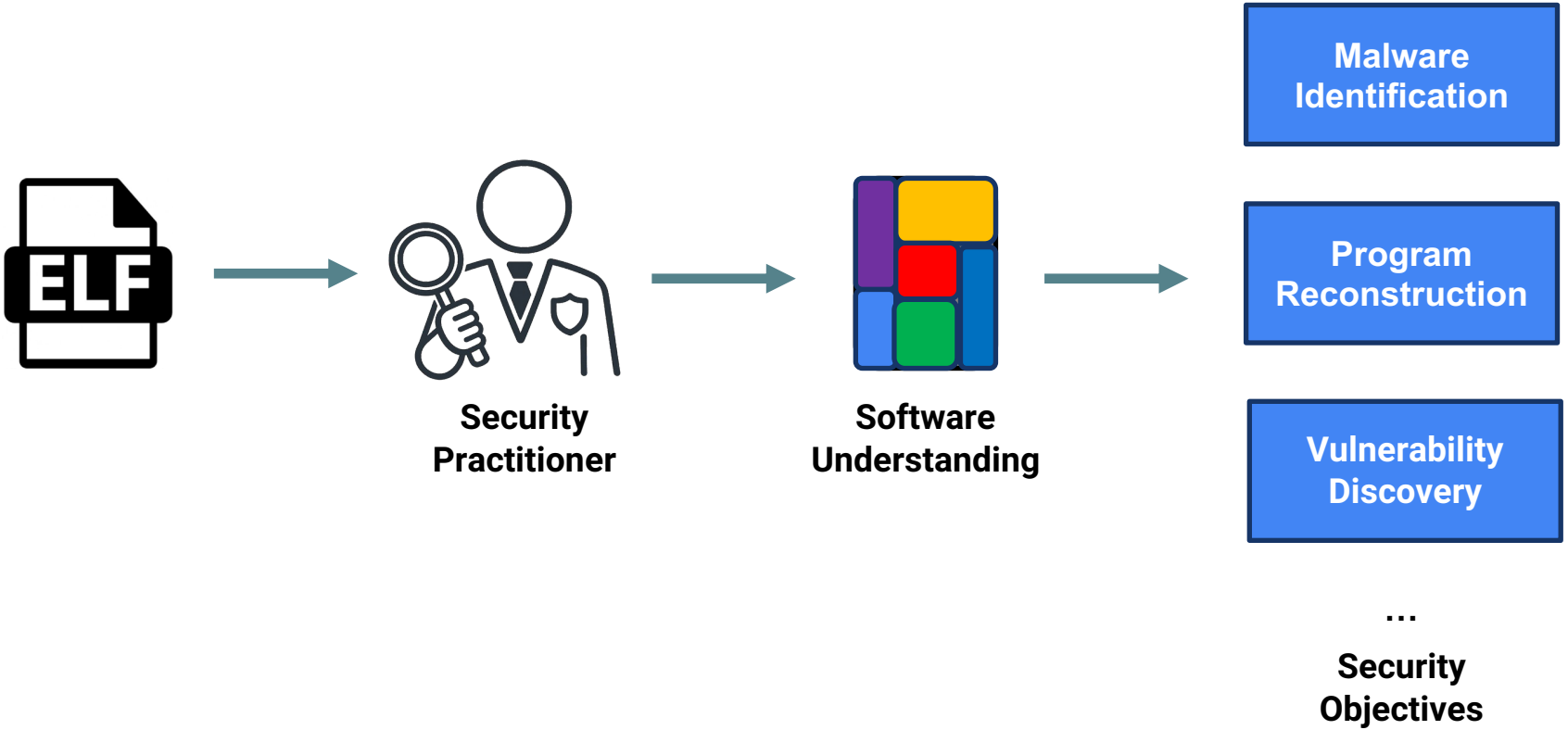
Securing Compiled Software

Security Practitioner

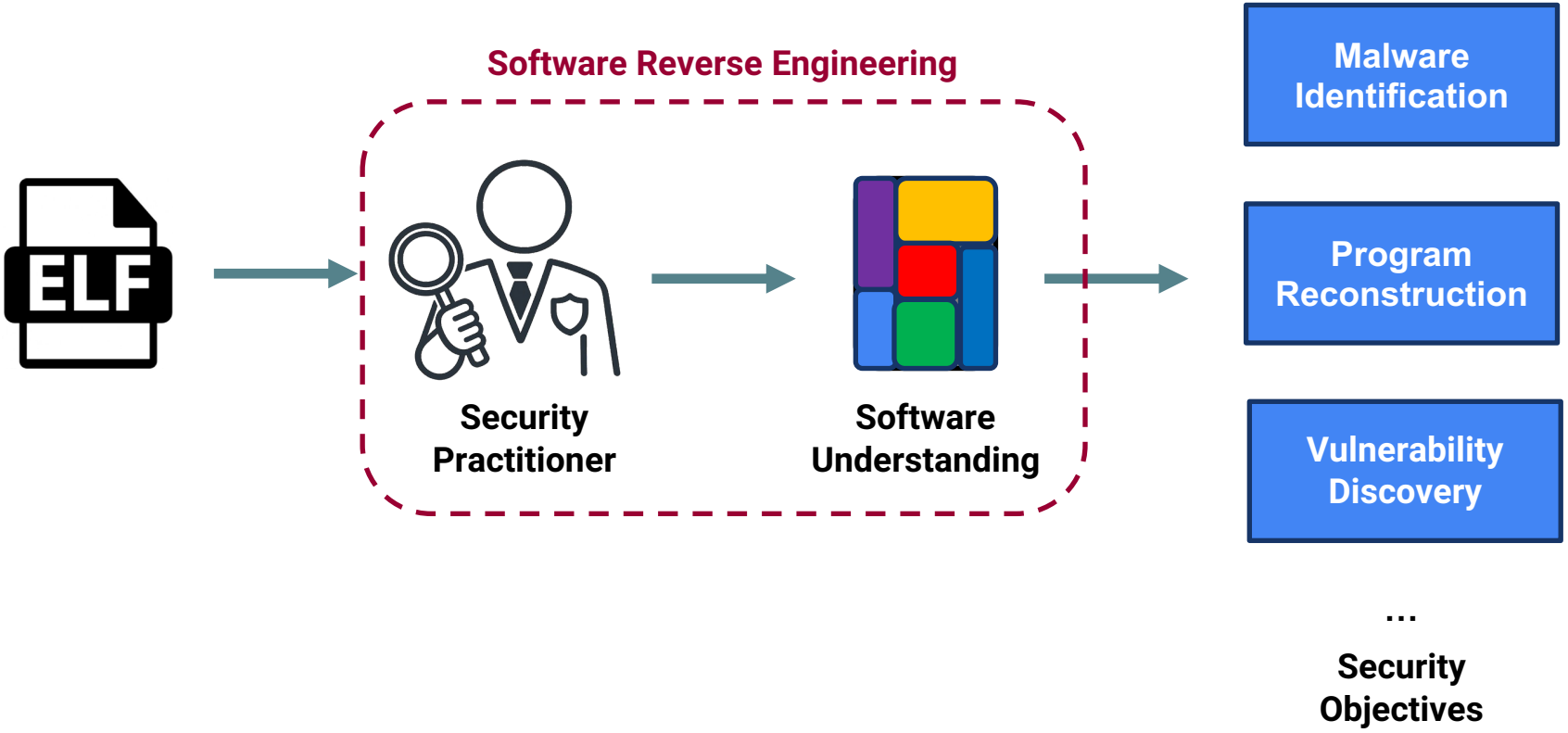
- Malware Identification
- Program Reconstruction
- Vulnerability Discovery
- ...

Security Objectives

Securing Compiled Software



Securing Compiled Software



Reverse Engineering (RE)

Human dominated [1]

1: Votipka, Daniel, et al. "An observational investigation of reverse Engineers' processes." *29th USENIX Security Symposium (USENIX Security 20)*. 2020.
2: Nosco, Timothy, et al. "The industrial age of hacking." *29th USENIX Security Symposium (USENIX Security 20)*. 2020.
3: Mantovani, Alessandro, et al. "RE-Mind: a first look inside the mind of a reverse engineer." *31st USENIX Security Symposium (USENIX Security 22)*. 2022.

Reverse Engineering (RE)

Human dominated [1]

Expensive (time and money) [2]

1: Votipka, Daniel, et al. "An observational investigation of reverse Engineers' processes." *29th USENIX Security Symposium (USENIX Security 20)*. 2020.
2: Nosco, Timothy, et al. "The industrial age of hacking." *29th USENIX Security Symposium (USENIX Security 20)*. 2020.
3: Mantovani, Alessandro, et al. "RE-Mind: a first look inside the mind of a reverse engineer." *31st USENIX Security Symposium (USENIX Security 22)*. 2022.

Reverse Engineering (RE)

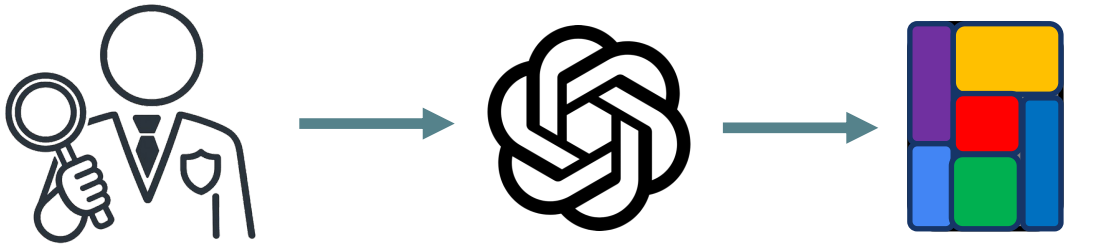
Human dominated [1]

Expensive (time and money) [2]

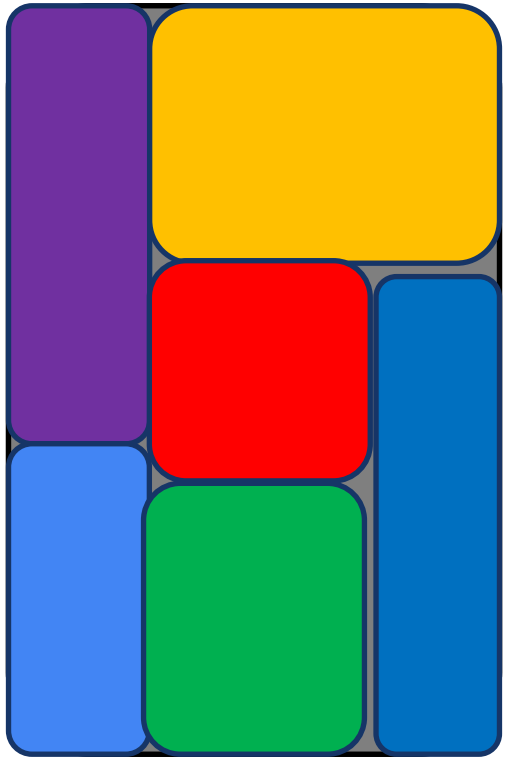
Expertise driven [3]

1: Votipka, Daniel, et al. "An observational investigation of reverse Engineers' processes." *29th USENIX Security Symposium (USENIX Security 20)*. 2020.
2: Nosco, Timothy, et al. "The industrial age of hacking." *29th USENIX Security Symposium (USENIX Security 20)*. 2020.
3: Mantovani, Alessandro, et al. "RE-Mind: a first look inside the mind of a reverse engineer." *31st USENIX Security Symposium (USENIX Security 22)*. 2022.

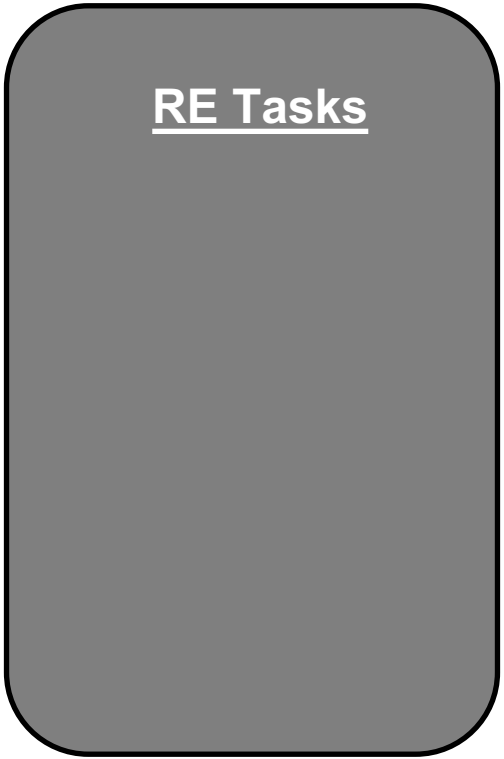
Modern Reverse Engineering: LLMs



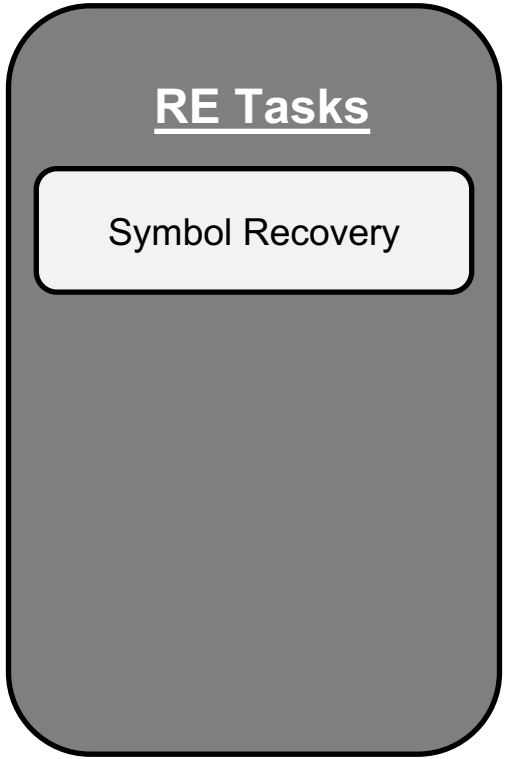
Reverse Engineering Tasks



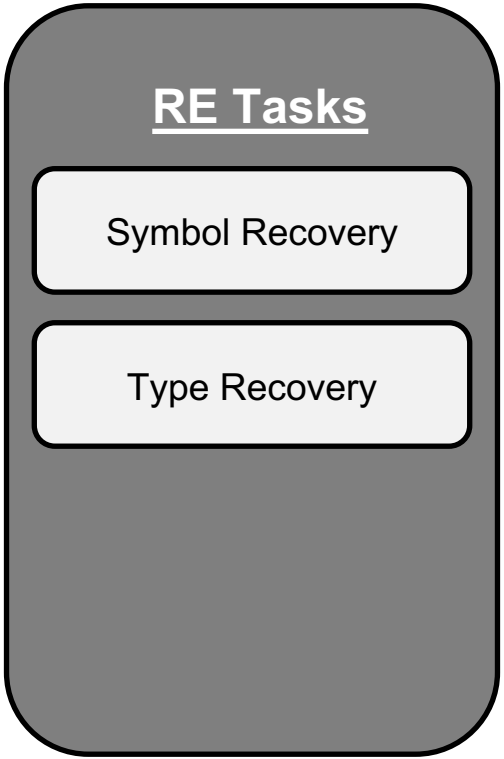
Reverse Engineering Tasks



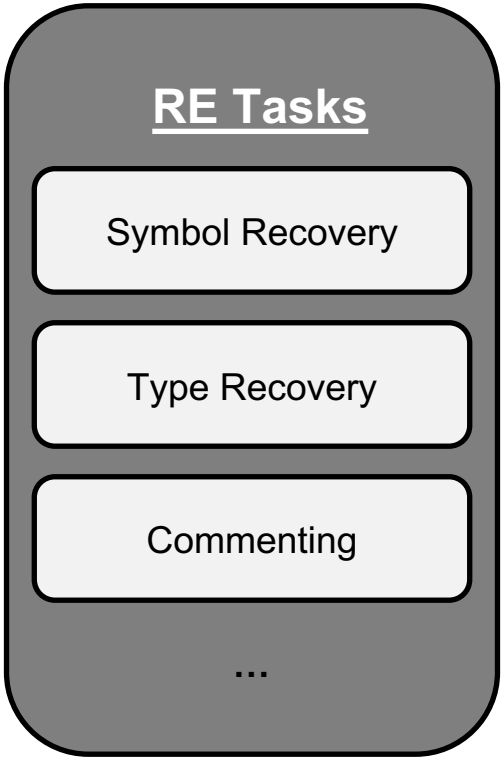
Reverse Engineering Tasks



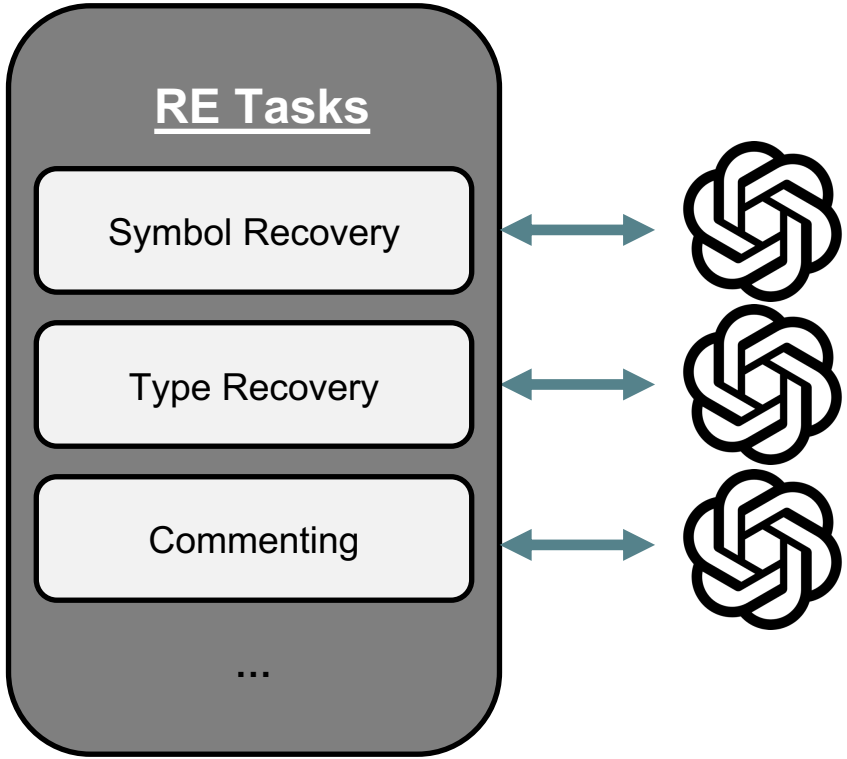
Reverse Engineering Tasks



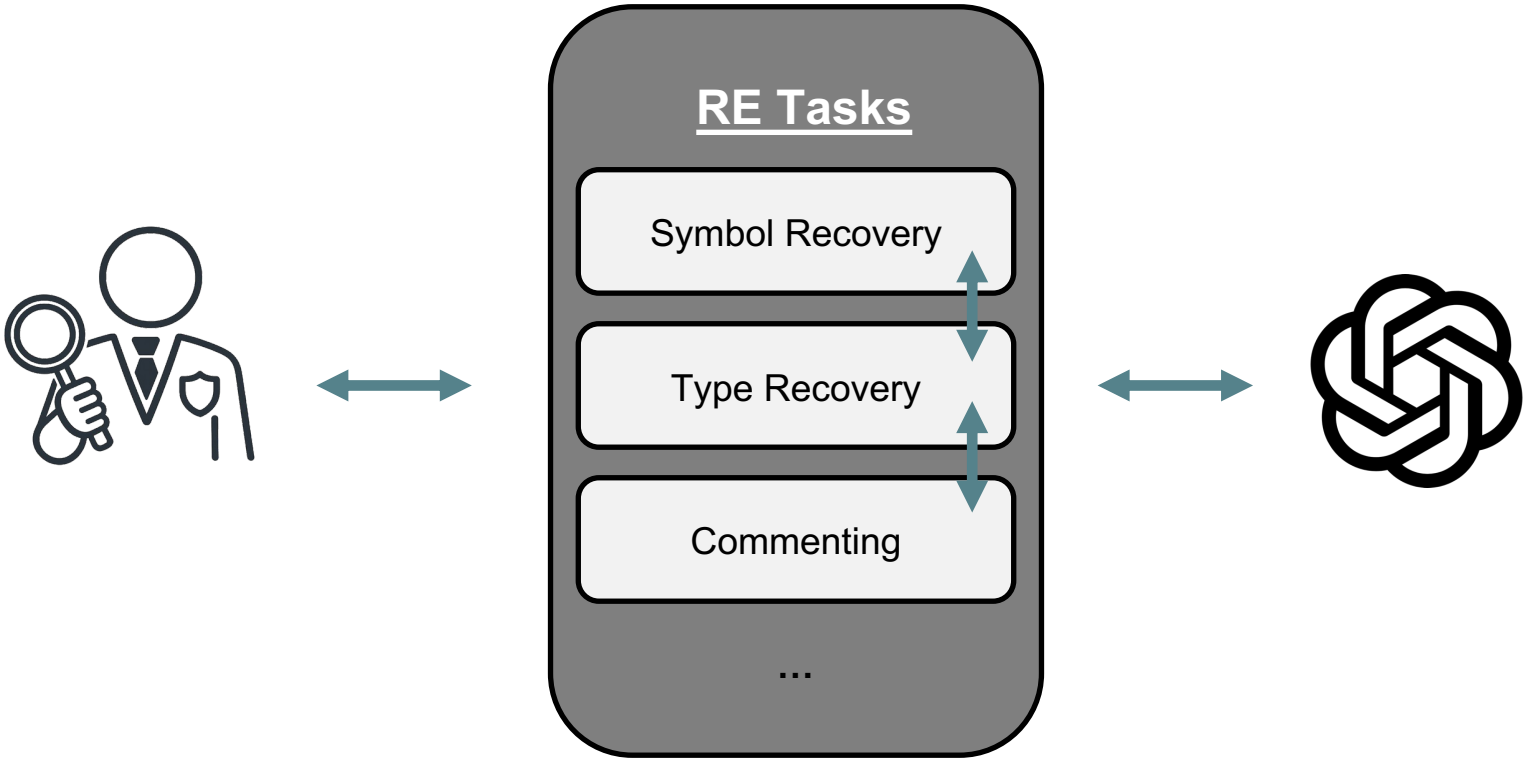
Reverse Engineering Tasks



Measuring LLMs for Specific RE Tasks



Our Work: Studying LLMs for End-to-End RE



Research Questions

How do practitioners **use** LLMs for RE?

How do LLMs **impact** RE performance?

Do LLM **collaboration** patterns impact RE performance?

Research Questions

How do practitioners **use** LLMs for RE?

How do LLMs **impact** RE performance?

Do LLM **collaboration** patterns impact RE performance?

Research Questions

How do practitioners **use** LLMs for RE?

How do LLMs **impact** RE performance?

Do LLM **collaboration** patterns impact RE performance?

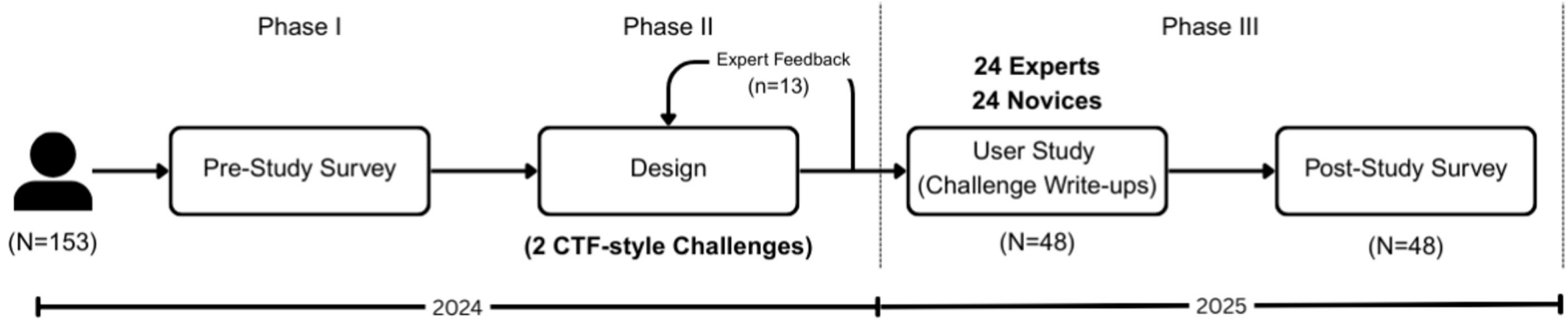
Research Questions

How do practitioners **use** LLMs for RE?

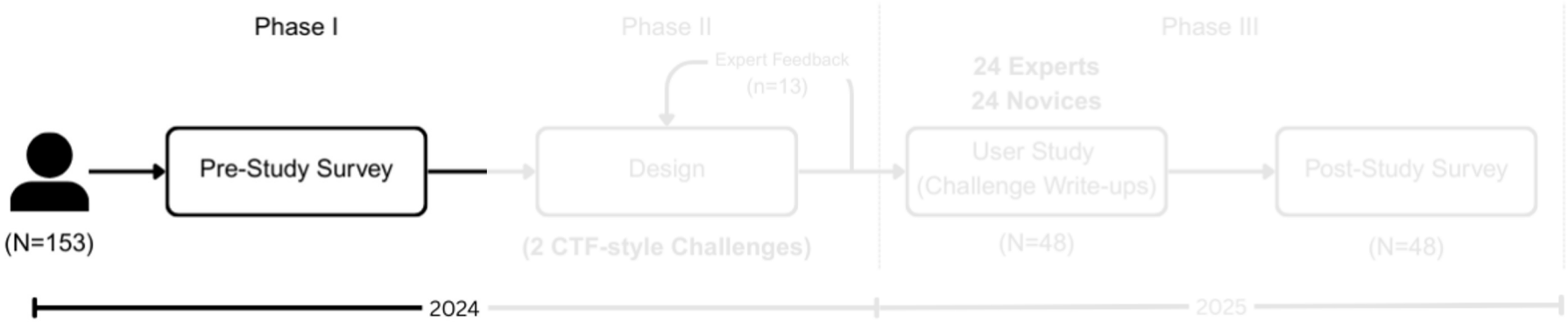
How do LLMs **impact** RE performance?

Do LLM **collaboration** patterns impact RE performance?

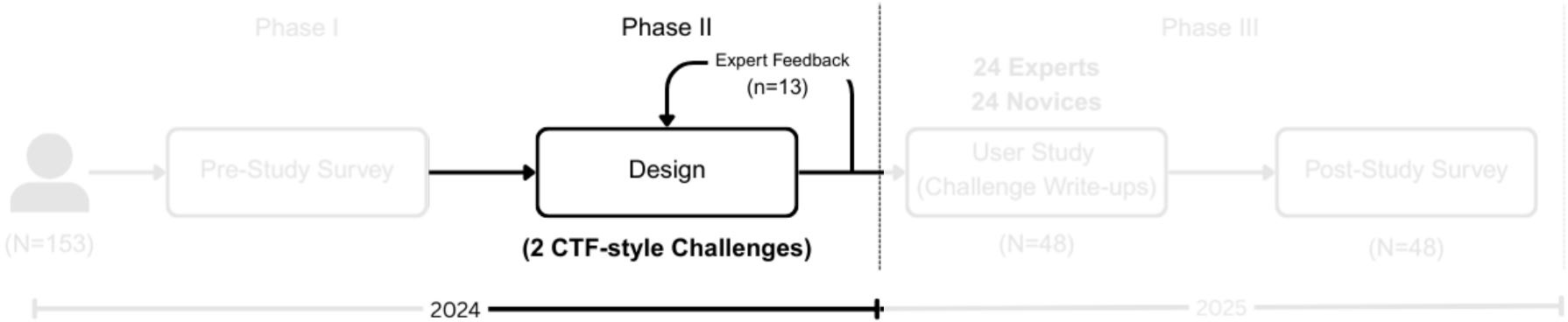
Study Design



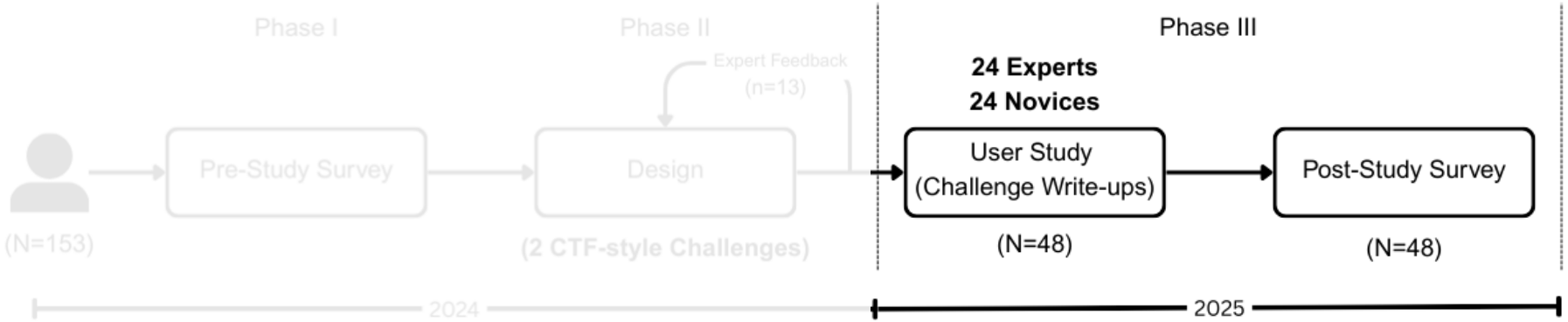
Study Design



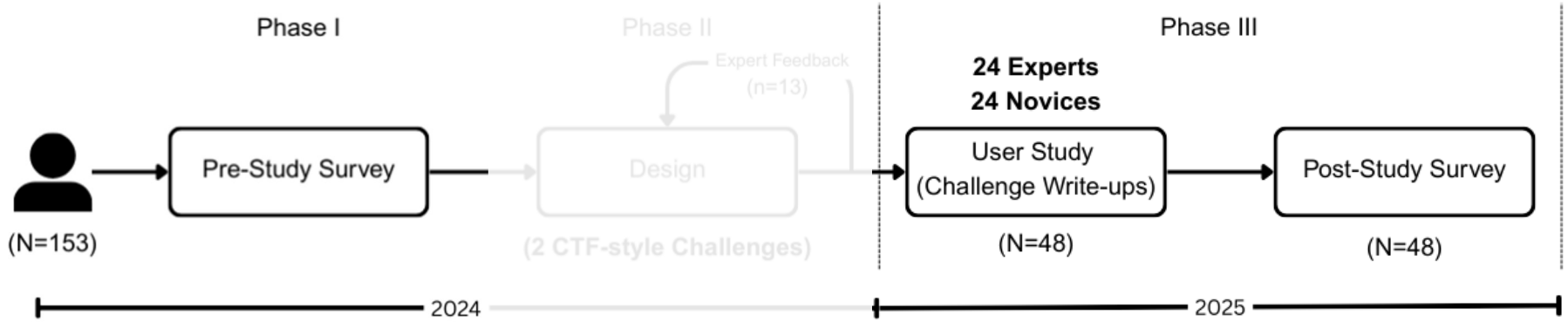
Study Design



Study Design



Study Design



Pre-Study Results

153 survey responses revealed **6** distinct applications of LLMs for RE with *decompilers*

Pre-Study Results

153 survey responses revealed **6** distinct applications of LLMs for RE with *decompilers*

- Function Summarization
- Function Identification
- Variable Symbol Recovery
- Function Symbol Recovery
- Vulnerability Identification
- Library Call Documentation

RE Experiment: Challenges

Two challenges inspired by CTFs to be reversed with decompilers

RE Experiment: Challenges

Two challenges inspired by CTFs to be reversed with decompilers

- An RPC server

RE Experiment: Challenges

Two challenges inspired by CTFs to be reversed with decompilers

- An RPC server
- An encrypted filesystem manager

RE Experiment: Challenges

Two challenges inspired by CTFs to be reversed with decompilers

- An RPC server
- An encrypted filesystem manager
- ~900 total lines of C

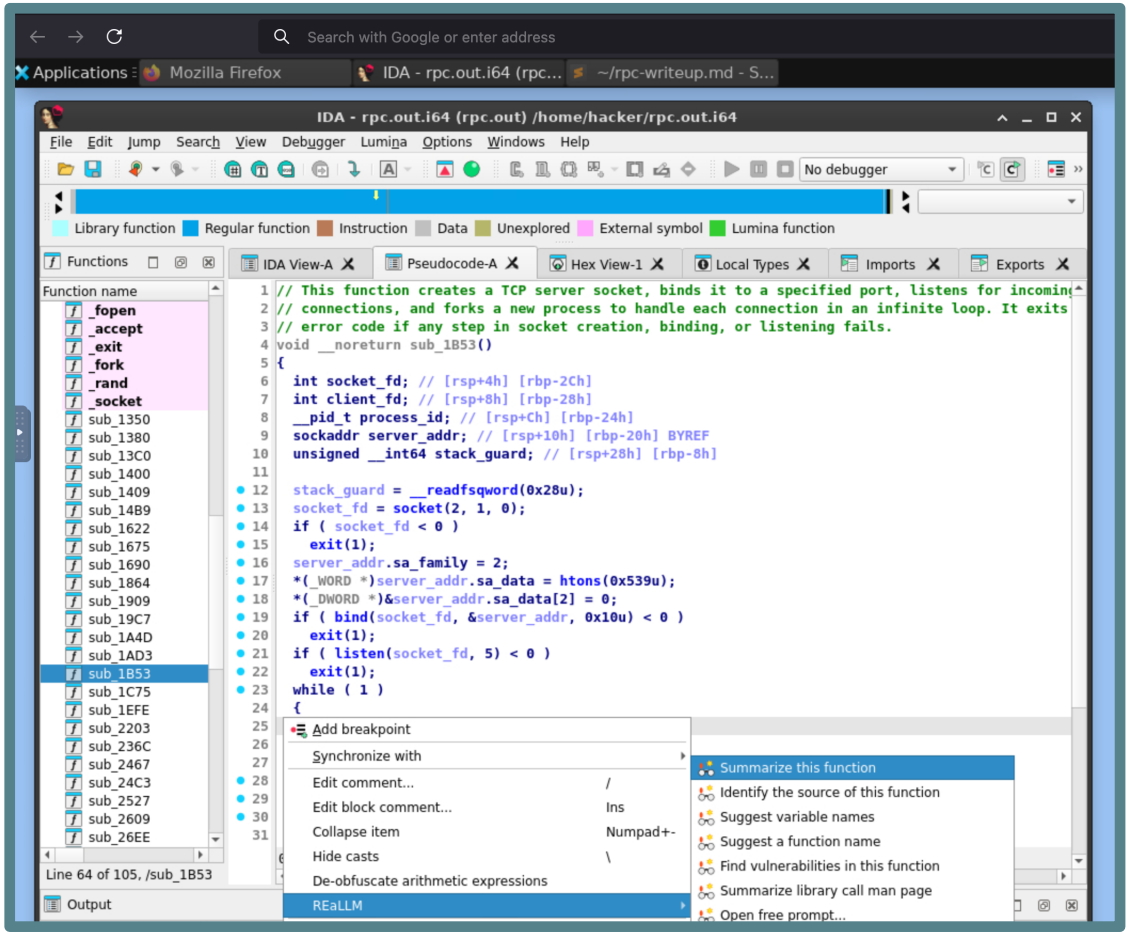
RE Experiment: Challenges

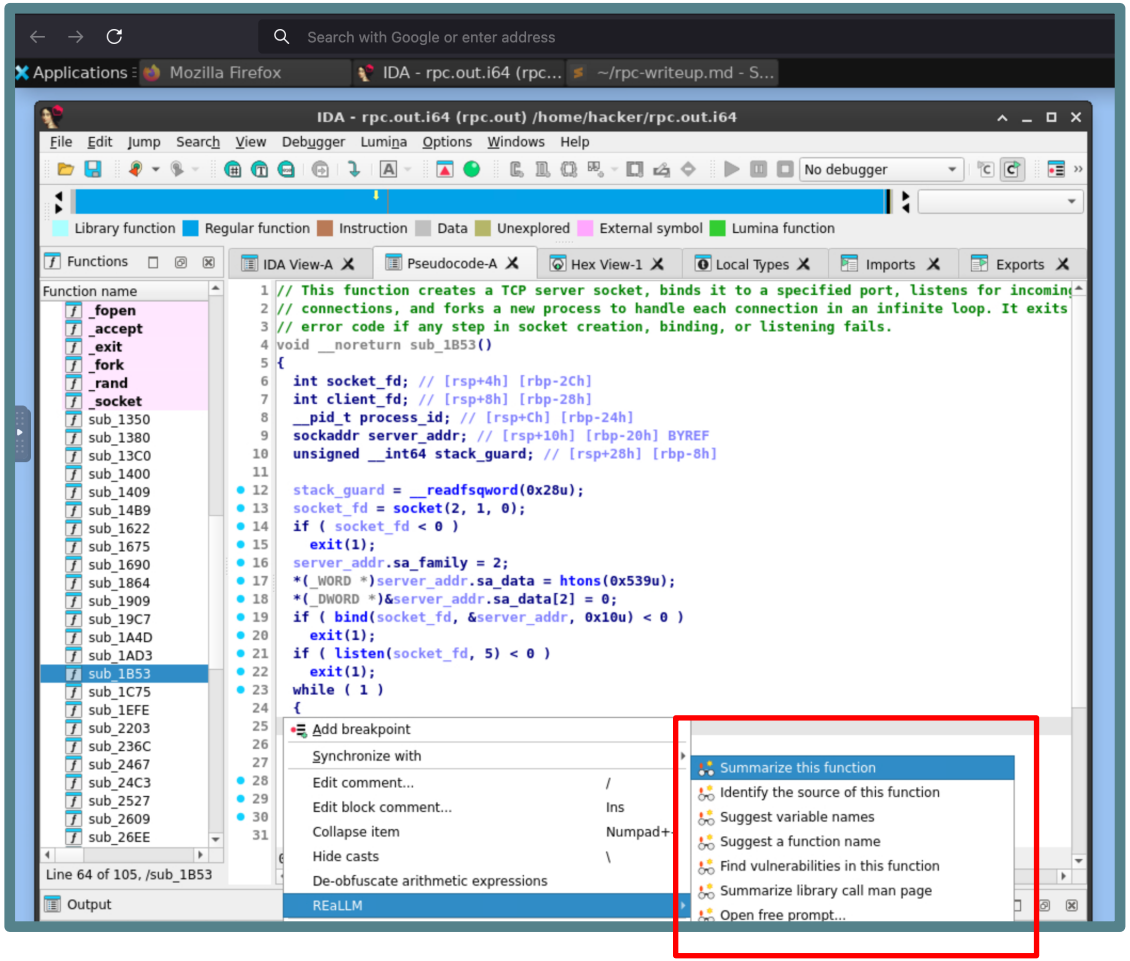
Two challenges inspired by CTFs to be reversed with decompilers

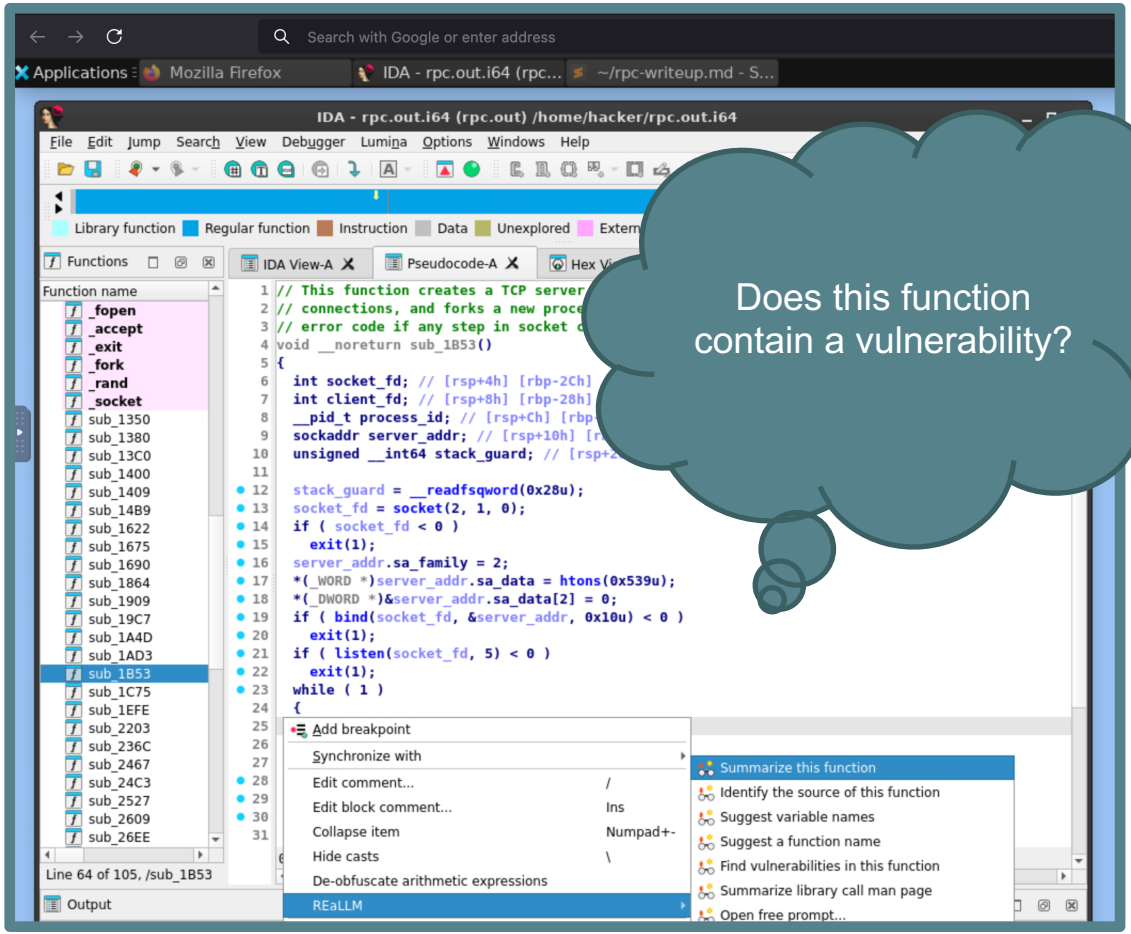
- An RPC server
- An encrypted filesystem manager
- ~900 total lines of C
- Compiled and stripped

RE Experiment: Workflow

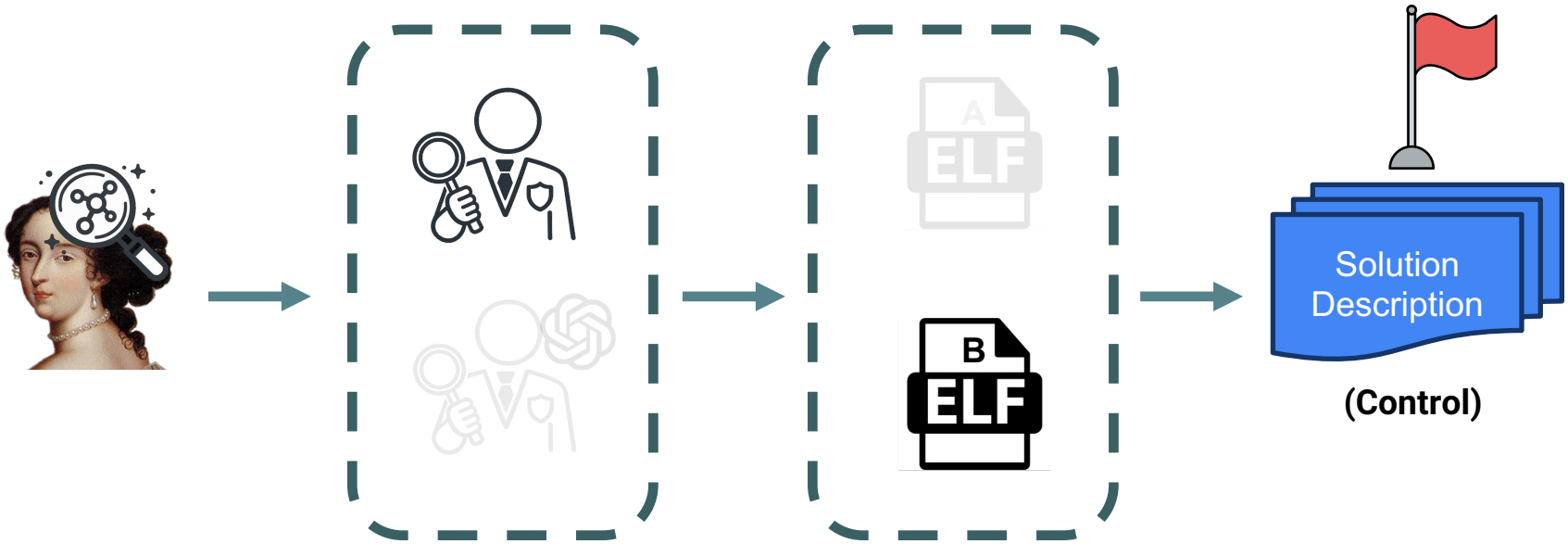
RE Experiment: Workflow


Search with Google or enter address
Applications
Mozilla Firefox
IDA - rpc.out.i64 (rpc...
~/rpc-writeup.md - S...
IDA - rpc.out.i64 (rpc.out) /home/hacker/rpc.out.i64
File Edit Jump Search View Debugger Lumina Options Windows Help
No debugger
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions
IDA View-A
Pseudocode-A
Hex View-1
Local Types
Imports
Exports
Function name
_fopen
_accept
_exit
_fork
_rand
_socket
sub_1350
sub_1380
sub_13C0
sub_1400
sub_1409
sub_14B9
sub_1622
sub_1675
sub_1690
sub_1864
sub_1909
sub_19C7
sub_1A4D
sub_1AD3
sub_1B53
sub_1C75
sub_1EFE
sub_2203
sub_236C
sub_2467
sub_24C3
sub_2527
sub_2609
sub_26EE
Line 64 of 105, /sub_1B53
// This function creates a TCP server socket, binds it to a specified port, listens for incomin
// connections, and forks a new process to handle each connection in an infinite loop. It exits
// error code if any step in socket creation, binding, or listening fails.
void __noreturn sub_1B53()
{
int socket_fd; // [rsp+4h] [rbp-2Ch]
int client_fd; // [rsp+8h] [rbp-28h]
__pid_t process_id; // [rsp+Ch] [rbp-24h]
sockaddr server_addr; // [rsp+10h] [rbp-20h] BYREF
unsigned __int64 stack_guard; // [rsp+28h] [rbp-8h]
stack_guard = __readfsqword(0x28u);
socket_fd = socket(2, 1, 0);
if (socket_fd < 0)
exit(1);
server_addr.sa_family = 2;
*(_WORD *)server_addr.sa_data = htons(0x539u);
*(_DWORD *)&server_addr.sa_data[2] = 0;
if (bind(socket_fd, &server_addr, 0x10u) < 0)
exit(1);
if (listen(socket_fd, 5) < 0)
exit(1);
while (1)
{
Add breakpoint
Synchronize with
Edit comment... /
Edit block comment... Ins
Collapse item Numpad+-
Hide casts \
De-obfuscate arithmetic expressions
REaLLM
Summarize this function
Identify the source of this function
Suggest variable names
Suggest a function name
Find vulnerabilities in this function
Summarize library call man page
Open free prompt...
Output

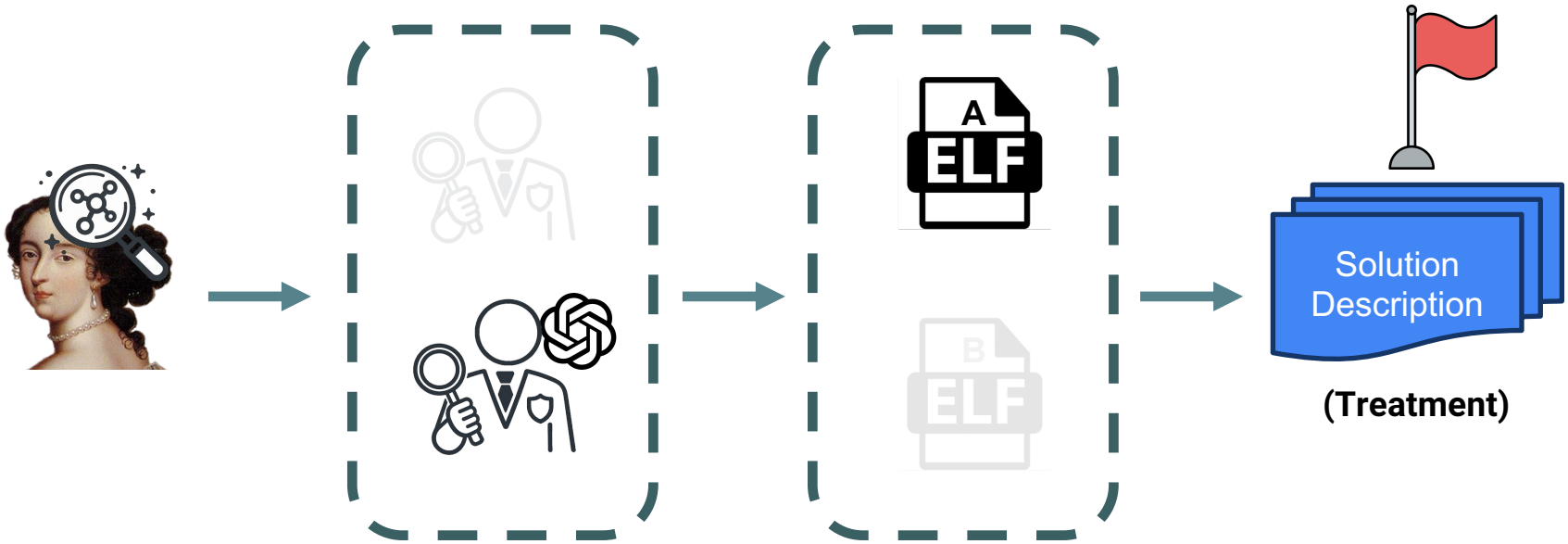

Search with Google or enter address
Applications
Mozilla Firefox
IDA - rpc.out.i64 (rpc...
~/rpc-writeup.md - S...
IDA - rpc.out.i64 (rpc.out) /home/hacker/rpc.out.i64
File Edit Jump Search View Debugger Lumina Options Windows Help
No debugger
Library function Regular function Instruction Data Unexplored External symbol Lumina function
Functions
IDA View-A
Pseudocode-A
Hex View-1
Local Types
Imports
Exports
Function name
_fopen
_accept
_exit
_fork
_rand
_socket
sub_1350
sub_1380
sub_13C0
sub_1400
sub_1409
sub_14B9
sub_1622
sub_1675
sub_1690
sub_1864
sub_1909
sub_19C7
sub_1A4D
sub_1AD3
sub_1B53
sub_1C75
sub_1EFE
sub_2203
sub_236C
sub_2467
sub_24C3
sub_2527
sub_2609
sub_26EE
Line 64 of 105, /sub_1B53
// This function creates a TCP server socket, binds it to a specified port, listens for incomin
// connections, and forks a new process to handle each connection in an infinite loop. It exits
// error code if any step in socket creation, binding, or listening fails.
void __noreturn sub_1B53()
{
int socket_fd; // [rsp+4h] [rbp-2Ch]
int client_fd; // [rsp+8h] [rbp-28h]
__pid_t process_id; // [rsp+Ch] [rbp-24h]
sockaddr server_addr; // [rsp+10h] [rbp-20h] BYREF
unsigned __int64 stack_guard; // [rsp+28h] [rbp-8h]
stack_guard = __readfsqword(0x28u);
socket_fd = socket(2, 1, 0);
if (socket_fd < 0)
exit(1);
server_addr.sa_family = 2;
*(_WORD *)server_addr.sa_data = htons(0x539u);
*(_DWORD *)&server_addr.sa_data[2] = 0;
if (bind(socket_fd, &server_addr, 0x10u) < 0)
exit(1);
if (listen(socket_fd, 5) < 0)
exit(1);
while (1)
{
Add breakpoint
Synchronize with
Edit comment... /
Edit block comment... Ins
Collapse item Numpad+
Hide casts \
De-obfuscate arithmetic expressions
REaLLM
Summarize this function
Identify the source of this function
Suggest variable names
Suggest a function name
Find vulnerabilities in this function
Summarize library call man page
Open free prompt...
Output


Search with Google or enter address
Applications
Mozilla Firefox
IDA - rpc.out.i64 (rpc...
~/rpc-writeup.md - S...
IDA - rpc.out.i64 (rpc.out) /home/hacker/rpc.out.i64
File Edit Jump Search View Debugger Lumina Options Windows Help
Library function
Regular function
Instruction
Data
Unexplored
Functions
IDA View-A
Pseudocode-A
Function name
_fopen
_accept
_exit
_fork
_rand
_socket
sub_1350
sub_1380
sub_13C0
sub_1400
sub_1409
sub_14B9
sub_1622
sub_1675
sub_1690
sub_1864
sub_1909
sub_19C7
sub_1A4D
sub_1AD3
sub_1B53
sub_1C75
sub_1EFE
sub_2203
sub_236C
sub_2467
sub_24C3
sub_2527
sub_2609
sub_26EE
Line 64 of 105, /sub_1B53
// This function creates a TCP server
// connections, and forks a new proce
// error code if any step in socket c
void __noreturn sub_1B53()
{
int socket_fd; // [rsp+4h] [rbp-2Ch]
int client_fd; // [rsp+8h] [rbp-28h]
__pid_t process_id; // [rsp+Ch] [rbp
sockaddr server_addr; // [rsp+10h] [r
unsigned __int64 stack_guard; // [rsp+2
stack_guard = __readfsqword(0x28u);
socket_fd = socket(2, 1, 0);
if (socket_fd < 0)
exit(1);
server_addr.sa_family = 2;
*(_WORD *)server_addr.sa_data = htons(0x539u);
*(_DWORD *)&server_addr.sa_data[2] = 0;
if (bind(socket_fd, &server_addr, 0x10u) < 0)
exit(1);
if (listen(socket_fd, 5) < 0)
exit(1);
while (1)
{
Add breakpoint
Synchronize with
Edit comment... /
Edit block comment... Ins
Collapse item Numpad+-
Hide casts \
De-obfuscate arithmetic expressions
REaLLM
Summarize this function
Identify the source of this function
Suggest variable names
Suggest a function name
Find vulnerabilities in this function
Summarize library call man page
Open free prompt...
Output
Does this function contain a vulnerability?

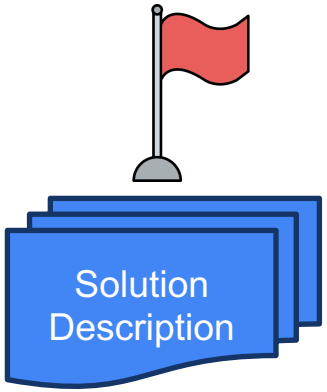
RE Experiment: Control



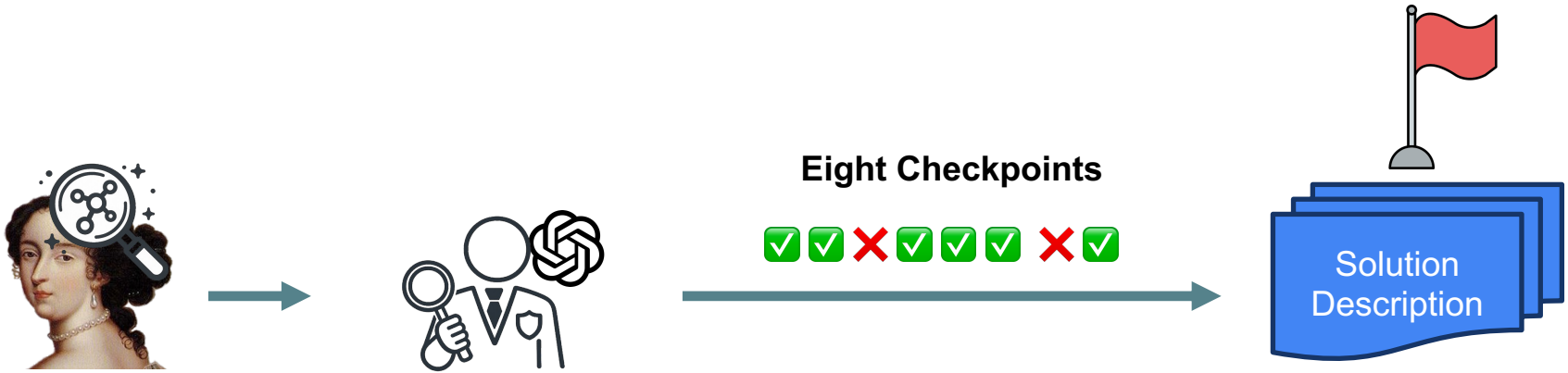
RE Experiment: Treatment



RE Experiment: Scoring



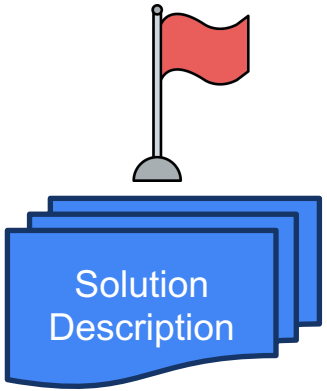
RE Experiment: Scoring



RE Experiment: Scoring

Eight Checkpoints

✅✅❌✅✅✅❌✅



Software Understanding Rate: ✅ Checkpoints / Time

Experiment Overview

Experiment Overview

48 Participants: 24 Experts, 24 Novices

Experiment Overview

48 Participants: 24 Experts, 24 Novices

109 Hours of recorded reverse engineering

Experiment Overview

48 Participants: 24 Experts, 24 Novices

109 Hours of recorded reverse engineering

1,517 LLM Queries

- Experts: 728
- Novices: 789

Experiment Overview

48 Participants: 24 Experts, 24 Novices

109 Hours of recorded reverse engineering

1,517 LLM Queries

- Experts: 728
- Novices: 789

Results: Closing The Expertise Gap

Results: Closing The Expertise Gap

Novices approached expert levels of reversing with a **2x increase** in understanding rates while using LLMs.

Results: Closing The Expertise Gap

Novices approached expert levels of reversing with a **2x increase** in understanding rates while using LLMs.

- Often due to function summarization.

```
__int64 __fastcall sub_2609(unsigned int a1, unsigned int *a2,
__int64 a3)
{
  unsigned int v4; // [rsp+20h] [rbp-18h]
  unsigned int v5; // [rsp+24h] [rbp-14h]
  unsigned int v6; // [rsp+28h] [rbp-10h]
  unsigned __int64 i; // [rsp+30h] [rbp-8h]

  v4 = *a2;
  v5 = a2[1];
  v6 = -1640531527 * a1;
  for ( i = 0; i < a1; ++i )
  {
    v5 -= (((v4 >> 5) ^ (16 * v4)) + v4) ^ (*(_DWORD *)(4LL *
((v6 >> 11) & 3) + a3) + v6);
    v6 += 1640531527;
    v4 -= (((v5 >> 5) ^ (16 * v5)) + v5) ^ (*(_DWORD *)(4LL *
(v6 & 3) + a3) + v6);
  }
  *a2 = v4;
  a2[1] = v5;
  return v5;
}
```

```
__int64 __fastcall sub_2609(unsigned int a1, unsigned int *a2,
__int64 a3)
{
  unsigned int v4; // [rsp+20h] [rbp-18h]
  unsigned int v5; // [rsp+24h] [rbp-14h]
  unsigned int v6; // [rsp+28h] [rbp-10h]
  unsigned __int64 i; // [rsp+30h] [rbp-8h]

  v4 = *a2;
  v5 = a2[1];
  v6 = -1640531527 * a1;
  for ( i = 0; i < a1; ++i )
  {
    v5 -= (((v4 >> 5) ^ (16 * v4)) + v4) ^ (*(_DWORD *)(4LL *
((v6 >> 11) & 3) + a3) + v6);
    v6 += 1640531527;
    v4 -= (((v5 >> 5) ^ (16 * v5)) + v5) ^ (*(_DWORD *)(4LL *
(v6 & 3) + a3) + v6);
  }
  *a2 = v4;
  a2[1] = v5;
  return v5;
}
```

“This function decrypts one 64-bit block using TEA (Tiny Encryption Algorithm).”

Results: Low Expert Improvement

Experts had **negligible** reversing improvements with LLMs.

Results: Low Expert Improvement

Experts had **negligible** reversing improvements with LLMs.

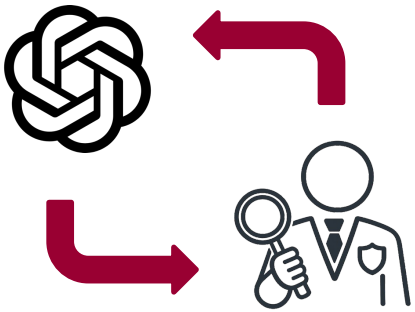
- Same LLM usage as novices, but for less benefit.

Results: Low Expert Improvement

Experts had **negligible** reversing improvements with LLMs.

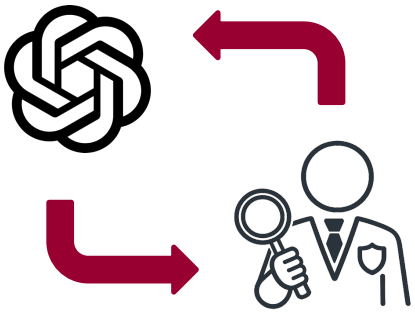
- Same LLM usage as novices, but for less benefit.
- Experts may require a **different collaboration model.**

Results: Wasted Collaboration



Results: Wasted Collaboration

Collaborating more on single functions with LLMs resulted in **lower understanding rates** on average.



Results: First Look Prevails

Most LLM-based understanding rate improvements happened in the **first seconds** of seeing a function.

Results: First Look Prevails

Most LLM-based understanding rate improvements happened in the **first seconds** of seeing a function.

- Compressible information either compressed quickly or never did.

Results: Hallucinations Remain A Risk

1,517 LLM Responses

3 detected hallucinations (<1%)

Results: Hallucinations Remain A Risk

1,517 LLM Responses
3 detected hallucinations (<1%)

3 vulnerability hallucinations caused…

- Caused a minimum **231% understanding rate slowdown**

Results: Hallucinations Remain A Risk

1,517 LLM Responses
3 detected hallucinations (<1%)

3 vulnerability hallucinations caused…

- Caused a minimum **231% understanding rate slowdown**
- All participants encountering it **submitted an incorrect solution early**!

Summary

We conducted the first empirical study on software reverse engineering with LLMs.

Summary

We conducted the first empirical study on software reverse engineering with LLMs.

We created a methodology to quantify the understanding of software in reverse engineering illuminating **18 findings**.

Summary

We conducted the first empirical study on software reverse engineering with LLMs.

We created a methodology to quantify the understanding of software in reverse engineering illuminating **18 findings**.

We open-sourced our platform, challenges, and LLM tool used in our study.

Open Science

Online RE Human Study Platform:
https://github.com/mahaloz/dec-synergy-study

Maintained LLM Tool:
https://github.com/mahaloz/DAILA
680 GitHub Stars
~1000 PyPI installs a month

Questions?

Zion Leonahenahe Basque
Arizona State University

zbasque@asu.edu
https://zionbasque.com
https://github.com/mahaloz/dec-synergy-study
https://github.com/mahaloz/DAILA