

# ReViver: Trace-Free Memory Data Forensics Through Speculative Symbolic Execution

Pengfei Sun

4N6 Research Group

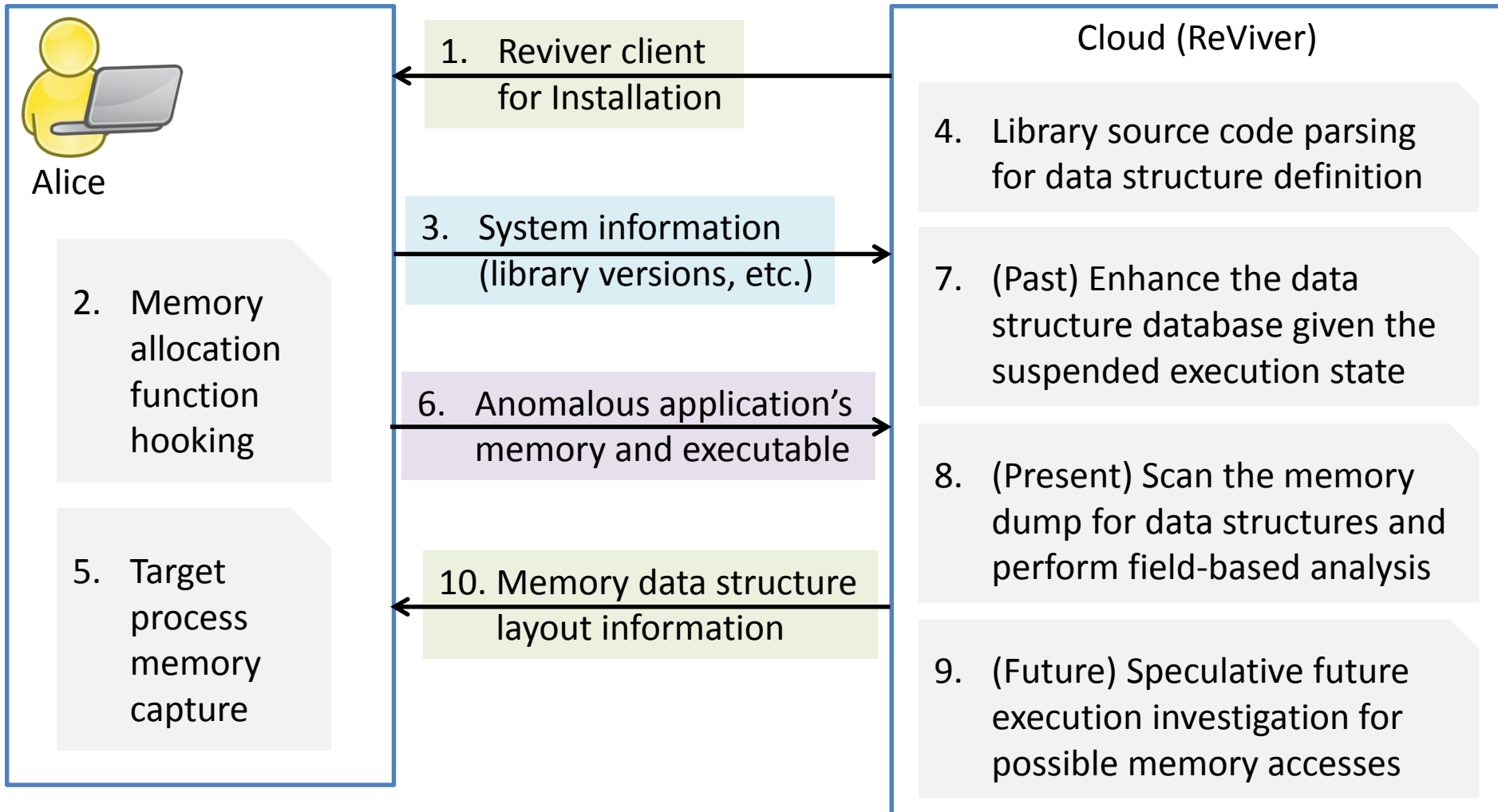
Rutgers University



- 300+ benign processes
  - misbehaving
  - root-cause
  - pre-unknown vulnerabilities



# ReViver's high level architecture



# Past: Statistical Information

## Goal:

- Collecting the statistical information about the executable's use of data structures

## How:

- Best-Effort Partial Symbolic Execution
- Prior Knowledge Collection
- Library Parsing for Structure Definitions

# Present: Static Heap Analysis

## Goal:

- Reversing data type **fine-grained** for captured memdump

## How:

- **Trace-free** heap reverse engineering
- **Pruning** candidate library structures
- Dump-library structure **matching**

# Future: Speculative Forensics

## Goal:

- Confirm or correct data structure by **type sinks**

## How:

- **Revived** speculative symbolic execution
- Type propagation for type forensics

# CoreUtils Groups Case Study

Part of Heap Memory Snapshot								Present	Past	Future	Past	
...												
09406ca0	0000	0000	0021	0000	6004	0940	6d00	0940	libname_list service_library	service_library	service_library	service_library
09406cb0	0000	0000	efef	efef	fefe	fefe	0018	0000				
...												
09407570	acfc	b76f	0e7f	af6f	efef	efef	fefe	fefe	known_function	known_function	known_function	known_function
09407580	0014	0000	0039	0000	0000	0000	0004	0000				
09407590	0018	0000	001b	0000	001e	0000	002e	0000	structure int[]	structure int[]	gid_t *	gid_t *
094075a0	006d	0000	007c	0000	03e8	0000	efef	efef				
094075b0	fefe	fefe	0030	0000	0000	0000	ea49	0001				
094075c0	0000	0000	0000	0000	0000	0000	0000	0000				
094075d0	0000	0000	0000	0000	0000	0000	0000	0000				
094075e0	0000	0000	0000	0000	0000	0000	0000	0000				
094075f0	0000	0000	a980	b76e	ffff	ffff	0041	0000				
09407600	0000	0000	0000	0000	7658	0940	ffff	ffff				
09407610	ffff	ffff	0000	0000	7664	0940	0000	0000				
09407620	0000	0000	ffff	ffff	ffff	ffff	0000	0000				
09407630	0000	0000	0000	0000	0000	0000	0000	0000	structure	structure	_IO_FILE	locked_FILE
09407640	0000	0000	0000	0000	0000	0000	0000	0000				
09407650	0000	0000	9a20	b76e	0000	0000	ffff	ffff				
09407660	0000	0000	0000	0000	0000	0000	0000	0000				
09407670	0000	0000	0000	0000	0000	0000	0000	0000				
09407680	0000	0000	0000	0000	0000	0000	0000	0000				
09407690	0000	0000	0000	0000	0000	0000	0000	0000				
094076a0	0000	0000	0000	0000	0000	0000	0000	0000				
094076b0	0000	0000	0000	0000	0000	0000	0000	0000				
094076c0	0000	0000	0000	0000	0000	0000	0000	0000				
094076d0	0000	0000	0000	0000	0000	0000	0000	0000				
094076e0	0000	0000	98a0	b76e	efef	efef	fefe	fefe				
094076f0	016c	0000	e911	0001	0000	0000	0000	0000				

exit: realloc  
ret: 0x09407588  
enter: getgrouplist (0xbfdcc424, 0x3e8, 0x9407588, 0xbfdcbcdc)  
definition: getgrouplist (const char \*, gid\_t, gid\_t \*, int \*)

Size:  
0x094075c0: 0x160  
\_IO\_FILE: 0x94  
locked\_FILE: 0x160

Probability:  
P(locked\_FILE) >  
P(\_IO\_FILE)

# Memory Data Forensics on Smart Grid SCADA System

- Learn about the memory layout and what data are available in memory
- Improve incident response
- Detect malicious modifications
- Figure out the root-cause of misbehaving

## Q&A