Targeted Memory Runtime Analysis

David Piché

May 16th, 2022

Polytechnique Montreal

DORSAL Laboratory

Agenda

1. Introduction
2. Related works
3. Our approach
4. System call tainted pointers
   - Problem
   - Solution
5. Demo
6. Results
7. Future Works

Introduction

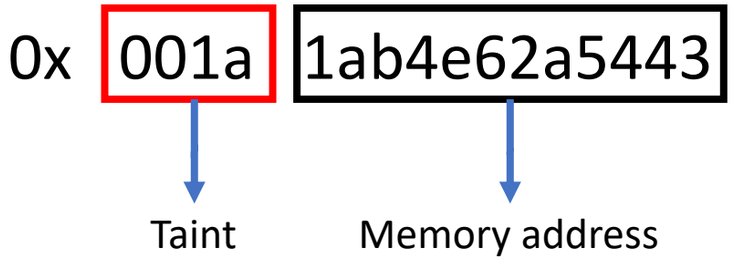
- Memory issues in C/C++ are still prevalent
  - Use-after-free
  - Memory leaks
  - Out-of-bound writes
  - And much more…

Related Works

- Adress Sanitizer
  - Uses shadow memory
  - Memory impact is too big for embarked systems
- Datawatch
  - Taint pointers stored in unused bits

Our approach

- X86_64 architecture
- Minimal approach to recreate datawatch:
  - Overwrite the *malloc/realloc* to add a taint.
  - Tainted pointers: use bits 47 to 63 for pointer tainting.



Our approach

- X86_64 architecture
- Minimal approach to recreate datawatch:
  - SIGSEGV as a handler to catch the tainted address.
  - For now, resume the flow of the program by removing the taint.

System Call Tainted Pointers: Problem

- System call arguments can be tainted.

- However, they are not handled by our SIGSEGV handler, as the system call is resolved in the kernel space

```
ptr =  taint(20, tag_data);

nW = write(1, ptr, sizeof(int)); // Not handled by our SIGSEGV handler
```

System Call Tainted Pointers: Solution

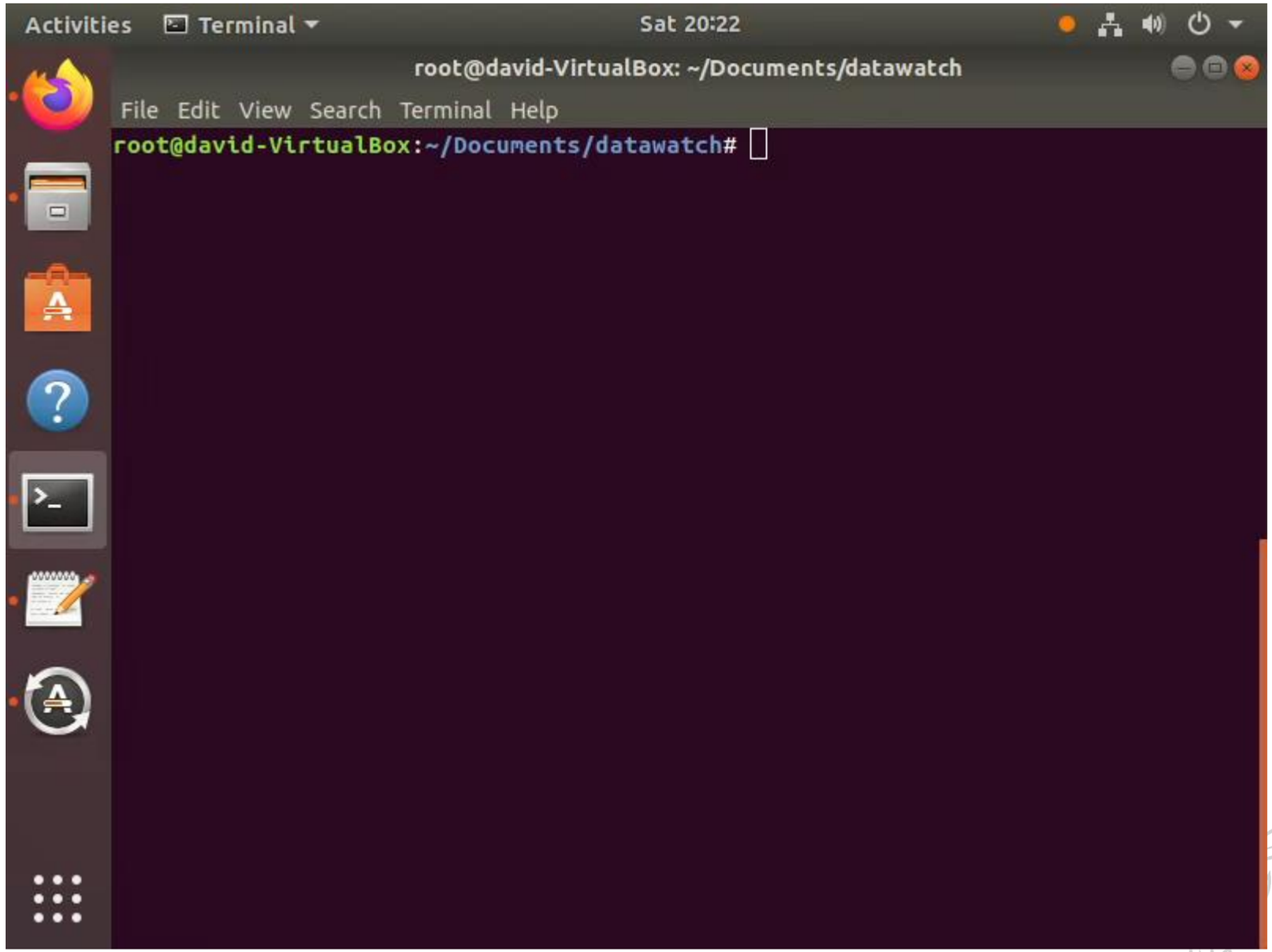
- Kernel patching

- Light modifications: 2 functions added, and calls to that function in 4 other files.

Demo

- Allocate a 4 byte pointer (using our malloc hook)
- Read/write using this tainted pointer as our argument
- Access the tainted pointer (generates a SIGSEGV signal)

Demo


Activities
Terminal
Sat 20:22
root@david-VirtualBox: ~/Documents/datawatch
File Edit View Search Terminal Help
root@david-VirtualBox:~/Documents/datawatch#

Results

- Tainting pointers will not occupy more space since it uses unused bits

- However, tainting pointers for every memory allocation will significantly slow down our program.

Future Works

- Use the upper 16 bits to store useful information:
  - Object id: identify objects more prone to memory errors
- **Targeted** memory analysis
- Bounds checking with Olivier's libpatch library