DD2497 - Project Specification
Group 2

Christopher Gauffin
Joseph Johansson
Jesper Lagnelöv
Kevin Harrison

November 11, 2022

1 Vulnerabilities countered by project

- Buffer overflows - Mainly we want to investigate how writing data outside of the bounds of a memory buffer to corrupt memory adjacent to that buffer can be used to manipulate the stack contents, what the consequences of that might be and what can be done to detect it.
- Memory corruption - We look mainly at corrupting memory of the stack.
- Code injections and Code-reuse - As a result of a corrupted memory, an overwritten return address on the stack can point to new malicious code or by reusing sensitive code that otherwise would require authorization.
- (Information leaks and side-channel attacks) - It might be interesting to further develop our solution to prevent leakage of sensitive information, more specifically making sure that the code is not susceptible to side-channel attacks.

2 Minimal requirements of project

- Create a dummy program that is susceptible to an buffer overflow attack.
- Demonstrate how the dummy program can be used to perform memory corruption and consequently a code injection and/or code-reuse attack.
- Develop detection software that successfully and deterministically prevent the attack. We call this Control Flow Manager (CFM).

    - Determine what data structure to use in order to store entries consisting of a function call, base pointer and return address.
    - Create the CFM server in the form of a driver or service that handles requests and makes changes to the data structure.
    - Find out what IPC calls to intercept in minix which will do the intermediate step of communicating with the CFM.
    - Be able to support multiple processes, i.e. there will be a collection of data structures, one for each process that are independent of each other, containing different entries. The CFM should be able to handle requests from the different processes and respond with an answer that fits the context of that specific process.

- Run the dummy program with the exploit simulatenously as we run the CFM and prove that it is working in real-time.

3 Optional requirements of project

- Show that stack canaries can be used but also demonstrate how they can be circumvented by an example.
- Implement minimal ASLR.
- Expand the detection software to also work against side-channel attacks.