Homework 2
MICS-204, Summer 2024

Karl-Johan Westhoff
email kjwesthoff@berkeley.edu

UC Berkleley School of Information
MICS Course 204 Summer 2024

CWE/SANS declared the top 25 most dangerous software weaknesses, 4 selected weaknesses

CWE-787 Out-of-bounds Write

Bjarne Stroustrup [1] once said: "C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off"[1]. Some programmers regard C/C++ as "high level" (People who code assembler and FORTRAN), others regards it as very "low level" (people who use python and js). Anyway, with languages where you get to directly access memory, there is a danger of putting data where it was not intended. To mitigate this code must be written carefully so the bits end up in the right place. For example:

- Check length before doing something to assure it is within what you have allocated room for
- use strncpy() instead of strcpy() (the first has a parameter for length of copied string so you can check it..)
- On the OS, "canaries" (places in memory which can be checked for overwrites) or deploy Address Space Layout Randomization (ASLR) which will reduce the risk of having malicious actors hit something that executes.

CWE-78 Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

"Never trust user inputs" Whenever something is used as inputs during execution, it must be ensured that creative formatting of the input does not get to run commands on the os. Possible ways are:

- Sanitize input for special characters that may be interpreted as commands on the OS
- Use abstraction, write pre written commands which are then selected based on user inputs - when possible (like using an ORM model for accessing databases)
- ReDoS attacks, where the user inputs are formed to crash the system matching strings using regular expressions, can be mitigated like above and additionally by limiting resources to the process, if excess resource consumption then it is probably malicious.

CWE-22 Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')

Is a variant of CWE 78 above. I remember being able to access folders from 'other departments' in Windows at work by using extra / / (relative path traversal). Mitigation is sanitation of inputs by removing consecutive "/¨s and ".."'s and combinations thereof.

CWE-807 Reliance on Untrusted Inputs in a Security Decision

Meaning that the inputs may have been manipulated by someone but are otherwise correct (MAC-spoofing, MITM attacks etc.) Mitigation here is to do something extra on a separate channel, for example 2 factor authentication.

[1] Created C++

"Make Least Privilege a Right (Not a Privilege)"

2.1 What are the five principles of least privilege (POLP) requirements? 2.2 What are the drawbacks of the chroot/jail approach? 2.3 What is one of the main difficulties with ad-hoc privilege separation? 2.4 How is a capability-based program similar to or different from allowlist?

"Memory corruption mitigation via hardening and testing"

3.1 Briefly describe the four exploit avenues mentioned in the paper. 3.2 Why code integrity cannot be fully enforced by browsers using Just-In-Time compilation? 3.3 Describe each step in the control-flow hijack exploit. For each step, discuss the mechanisms which could either detect or prevent the step. 3.4 Why data corruption attacks are called non-control-data attacks? Give one example of a non-control-data attack involving user identity data. 3.5 What are the most widely deployed protection mechanisms against memory corruption attacks? 3.6 Beside security, what is the most important requirement for protection mechanisms as stated by the paper? How does this requirement affect gaining wide adoption in production environments? 3.7 What is the novel protection mechanism proposed by the author to prevent control-flow hijack

References

[1] *Bjarne Stroustrup website*. `https://www.stroustrup.com/quotes.html`. Accessed: 2024-11-10.

Appendix