Software Security II

Friday, April 6, 2018 12:08 PM

Data Representations

Null Termination

- String: null-terminated?
- X.509 public key certificate standard allows for null characters in the Distinguished Name (DN)
- SSL implementation bug; browser:
  - Verifies cert based on non-null termination
  - Compares name by null termination
  - Browser incorrectly displays web address (even though certificate of spoofed website is valid)

UTF-8 Encoding

- Ascii "/" = 0010 1111 (2F)
- 2,3,4 byte version: have to check if-else multiple times
- File path system
  - Allows user to access files from certain directory
  - Have to check that "../" is not in user input
  - Checks may be incomplete, can mask "/" as %2F

IP Address

- Represented with string of ascii, separate with "." OR unsigned integers
- 137.132.2.6 -> 137.132.0.518
  - Verify with 4 integer version, but using 32-bit representation
- Guideline: use canonical (unique) representation (convert user input -> internal standard representation)

Buffer Overflow

- C++, C directly manage memory
- Depends on how compiler arranges memory, usually two variables are contiguous
- `strcpy in C`
  - Includes the terminating null character in copying (length + 1)
  - Over-printing compromises confidentially/ Segmentation Fault
  - Should use strncpy

Stack Overflow (Stack Smash)

- Stack overflow -> modified (can modify return address)
- Can inject attacker code into memory, control flow directs to malicious code
- Effective canaries to detect stack overflow

Integer Overflow

- Modular arithmetic

Code Injection

- Scripting languages: interpreted by another program at run time
  - Can be modified while interpreted
- Can also be injected via buffer overflow

SQL injection

- 'anything' OR 1=1--'
- Returns all rows

Undocumented Access Point

- Easter eggs, back door, logic bombs (unhappy programmers)

Race Condition (TOCTOU)

- Time-of-check-time-of-use
- Multiple processes access data, outcome depends on sequence of access
- A access data, B swaps data

***examples 1,2

Defence and Preventive measures

- Halting Problem: difficult to ensure programme is bug-free (may not terminate)

Input Validation (filtering)

- White list/ black list
- Difficult to be complete

Safe functions

- strncpy instead of strcpy
    - Still, strlen() + strncpy() is vulnerable
- printf(f), access()

Bound checks, type safety

- Halt program/ throw exception if out of bounds
- Reduced efficiency
- No bound checks in C, C++
- Type checking: 8-bit vs. 64-bit integer (Type Safety)
- Check dynamically (runtime) or statically (compile time, no inputs)

Canaries and Memory Protection

- Detect overflows, esp. stack overflow
- Overwriting memory has to be consecutive, protect location with canary in front
- Canary = 0: strcpy cannot copy null character
    - Value has to be secret

Memory Randomisation

- Address Space Layout Randomisation (ASLR)

Code Inspection

- Manual/ automated
- Taint analysis
    - Source: variables which can be influence by user
    - Sink: critical functions
    - Static/ dynamic

Testing

- White-box, black-box, grey-box: access to the code?
- Fuzzing: intentionally sends malformed inputs (need not be random inputs), can be automated

Principle of Least Privilege

- Give applications least privilege
- Client should not have the responsibility to harden the system by themselves

Patching

- Life cycle of vulnerability (low -> high -> low popularity)
  - Zero-day vulnerabilities: previously unknown, usually state-sponsored
- May lead to more problems due to patch problems
  - Patch Management

Readings

Null charac
https://tools.cisco.com/security/center/viewAlert.x?alertId=19157
SSL bug
https://www.ruby-lang.org/en/news/2013/06/27/hostname-check-bypassing-vulnerability-in-openssl-client-cve-2013-4073/
Null charac
https://security.stackexchange.com/questions/31760/what-are-those-nul-bytes-doing-in-certificate-subject-cn