Secure Software Programming and Vulnerability Analysis

Christopher Kruegel chris@auto.tuwien.ac.at
http://www.auto.tuwien.ac.at/~chris

Input Validation

Overview

- Systems receive data from variety of sources
  - from software's users to
  - remote systems on a network

- Often, these sources are untrusted
  - potentially hostile

- Every piece of data needs to be checked
  - data has to be as anticipated (conform to specification)

➢ Input Validation

Input Validation

- Task
  - determine legal input
  - clean input data from illegal parts (filtering)

- Filtering
  - use principle of “fail-safe defaults”
  - reject anything that is not explicitly considered legal, don’t specify filters for “bad” input
  - “bad” input is only useful to test filter

  - check for data content and data length (minimum and maximum length)

Input Validation

- Drop privileges while parsing input
  - especially when using unsafe languages (e.g., C / C++) and
  - complex parsers such as `lex` and `yacc`

- Use trustworthy channels
  - especially for authentication and pre-validated input

- Deputy problem
  - checking program and program that uses data make slightly different assumptions
  - standards are helpful, but often specific extensions

Input Validation

- Common sources of untrusted input
  - Input that a program operates on (local or remote)
    - strings
    - files
    - web-based transactions (HTML, URLs, session management)
  - Input that controls program behavior
    - command line arguments
    - environment variables
    - configuration files
    - locale settings
    - signals

Strings

- Optimally, specified as regular expression

- Problem
  - control and metacharacters

- Metacharacters
  - characters that are not interpreted as data
  - used as delimiters or command characters
  - Examples
    - command line shell
    - SQL interpreter
    - terminals
  - WWW Security FAQ [Stein 1999]

```
& ; ` ' \ " | * ? ~ < > ^ ( ) [ ] { } $ \n \r
```

Strings

- Metacharacters
  - shell is used for several important library calls
    `popen`, `system`
  - metacharacters need to be escaped

- Line ending encoding

Platform	Line Encoding	ASCII
Apple	[CR]	0x0d
UNIX	[LF]	0x0a
DOS / Windows	[CR][LF]	0x0d 0x0a
OS/390	[NEL]	0x85

  - HTTP specification
    - ISS evasion attack

Character Encoding

- Strings are represented as characters

- Traditionally, 8-bit ASCII characters were used
  - only 256 characters possible
  - unsuitable for many languages except English

- ISO 10646 Universal Multiple-Octet Coded Character Set (UCS)
  - unique 31 bit values for each character
  - first 65536 characters termed16-bit BSM (basic multi-lingual plane)
  - merged with Unicode forum efforts

- Problem with existing programs that expect a character to be a byte

➢ UTF-8 encoding

Character Encoding

- UTF-8 encoding
  - variable length encoding (character is 1 to 7 bytes long)
  - classical US ASCII characters (0 to 0x7f) encode as themselves
  - UCS characters beyond 0x7f are encoded as a multi-byte sequence consisting only of bytes in the range 0x80 to 0xfd
    - especially, no null character permitted
  - lexicographic sorting order of UCS-4 strings is preserved
    - good for search algorithms

- Problem
  - same value can be encoded in different ways
  - standard now requires “smallest possible form”
  - thus some sequences are not permitted
  - opens problems for misinterpretation

    `00` could be encoded (illegally) as `C0 80`

Files

- File names
  - avoid globbing (i.e., expanding metacharacters such as *)
  - avoid directory delimiter ‘`/`’ and control characters ‘`..`’
  - problematic characters
    - ‘`\0`’
      end of filename
    - ‘`-`’
      misinterpreted as program argument
      filename “`-la`”
      `ls *` gets expanded to `ls -la <other files>`
    - control characters
    - spaces
      - separation of arguments

Files

- File descriptors
  - invoking program chooses file descriptors
    - used for redirection
  - Problem
    - standard file descriptors (`stdin, stdout, stderr`) can be closed
    - next open operation opens smallest file descriptor
    - for example, when open reopens `stdout`
    - then all regular print statements are also sent to this file
- File contents
  - must be protected properly (permission settings)

Command Line Arguments

- Obviously complete control of attacker
  - `execve` system call

- Important
  - program name can be tampered with
  - `argv[0]` cannot be trusted

Environment Variables

- Inherited from calling process
  - also complete control of attacker
  - should be cleared (or at least sanitized)
  - add safe default values
    - `PATH`, `IFS`, `TZ` (time zone)

- Dangerous environment variables
  - IFS (internal field separator)
    used to determine argument separator
    can be set to ‘/’
    system(“/bin/bash”) calls `bin` program in local directory

Environment Variables

- Dangerous environment variables storage format
  - array of pointers to strings
  - strings have “`name=value`” format
  - possible to have set up multiple entries of same name
  - checked and used variable might be different

- User control over environment
  - via configuration files (e.g., user login file, `.ssh/environment`)
  - using protocol support (telnet environment option)
  - should be avoided

    `LD_PRELOAD` attack against `nologin` program

Locale Settings

- Combination of language and cultural factors
  - internationalization (support for multiple locales)
  - localization (usage of a particular locale)

- Selection
  - for local programs via environment variables
  - for web applications via browser request line

- Support
  - using the `catgets` or `gettext` interface
  - `catgets` uses integer indexes into string tables
  - `gettext` uses a mapping from English text to the locale
  - heavily dependent on environment variable settings

Web Issues

- Web traffic is ubiquitous
  - few popular web servers
  - but many (custom made) web applications
- Many security concerns
  - complex specifications
    - URL encoding
    - HTML
  - stateless protocol
    - session tracking needed
    - URL rewriting
    - hidden fields
    - cookies

CGI Programs

- Common Gateway Interface
  - programs run on trusted server
  - receive input from remote clients via `stdin` and environment variables
- Often, script languages are used
  - shell code
  - Perl
  - PHP
- Unchecked input is the biggest issue according to OWASP Top 10 list

CGI Programs

- Perl supports “tainting” mechanism
  - unsafe input is “tainted”
  - tainted input cannot be used in unsafe operations unless explicitly checked
  - check works by using regular expressions
  - run-time check
- PHP can be configured with unsafe “global register” behavior
  - all query variables are immediately copied into global variables
  - allows easy access to query variables in scripts
  - can unexpectedly overwrite any variable that is not initialized
- Forbid HTTP GET for non-queries
  - to transmit data, POST should be used
  - GET transmits data via URL
  - attacker can create malicious URLs that can be clicked or auto-loaded and perform undesired actions

URL Encoding

- Values in URLs can be URL-encoded
  - must be decoded properly
- Hex encoding (RFC compliant)
  - %XX, where XX is hexadecimal ASCII value of character
    A = %41
- Double hex encoding (Microsoft IIS)
  - %25XX, where XX is hexadecimal ASCII value of character (%25 = %)
    A = %25XX
- Double nibble hex encoding (Microsoft IIS)
  - each hexadecimal nibble is separately encoded
    A = %25%34%31

HTML Filtering

- Problem
  - HTML input data that is sent to other users
  - Cross-site malicious content
    - attacker embeds malicious content in page that is displayed at victim’s site
    - can be used to

      steal cookies, launch denial of service attacks (spawn windows, tamper with fonts, …), or display bogus forms
- Solution
  - Filter HTML tags
    - make sure to remove ‘<‘ and ‘>’
  - Encode HTML tags
    - using **&**val**;**

HTML Filtering

- When HTML data must be accepted
  - use validation of HTML data
  - list of “safe” HTML tags
  - nesting must be balanced
  - check attributes (some may contain scripts)
- Validating links (URIs/URLs)

```
URI = scheme://authority[path][?query][#fragment]
authority = [username[:password]@]host[:portnumber]
```

  - scheme should be restricted to `http` / `https`
  - most other options should be immediately removed (user / passwd)

HTTP Sessions

- Sessions are needed for applications that require state
  - all applications that require authentication
- Session ID can be
  - encoded in URL (caching, stored in referrer logs of other sites)
  - hidden fields (not all requests are POSTs)
  - cookies (preferable, but cookies can be disabled)
- Cookie
  - token that is stored on client machine
  - set by server
  - uses a single domain attribute
    - cookies are only sent back to servers whose domain attribute match

HTTP Sessions

- Non-persistent cookies
  - are only stored im memory during browser session
  - good for sessions
- Secure cookies
  - cookies that are only sent over encrypted (SSL) connections
- Only encrypting the cookie over insecure connection is useless
  - attackers can simply replay a stolen, encrypted cookie
- Cookies that include the IP address
  - makes cookie stealing harder
  - breaks session if IP address of client changes during that session