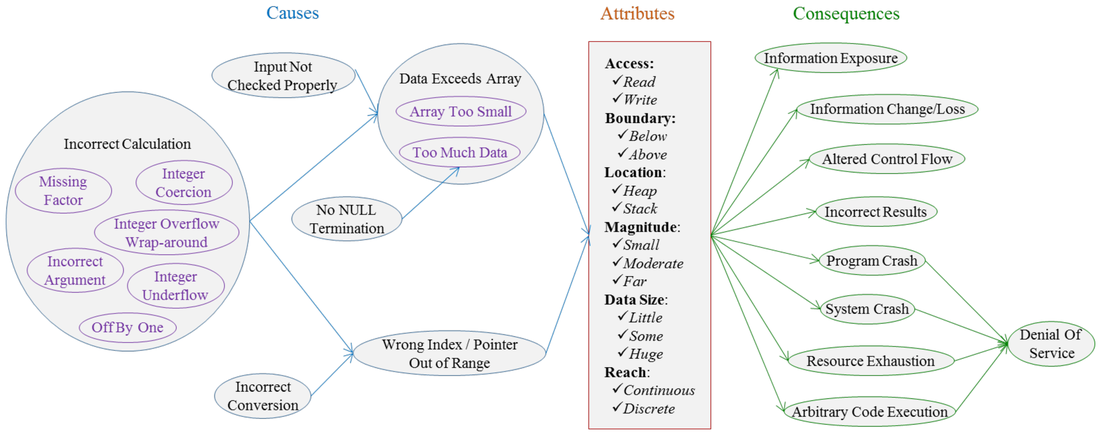
The Bugs Framework (BF): A Structured, Integrated Framework to Express Software Bugs

Irena Bojanova, NIST; Paul Black, NIST; Yaacov Yesha, NIST, UMBC; Yan Wu, BGSU
{irena.bojanova, paul.black, yaacov.yesha}@nist.gov, yayesha@cs.umbc.edu, yanwu@bgsu.edu

To achieve higher levels of assurance for digital systems, we need to answer questions such as does this software have bugs of these critical classes? Do two software assurance tools find the same set of bugs or different, complimentary sets? Can we guarantee that a new technique discovers all problems of this type? To answer such questions, we need a vastly improved way to describe classes of vulnerabilities and chains of failures. We present a descriptive Bugs Framework (BF) that will raise the current realm of best efforts and useful heuristics. We provide definitions of three weakness classes, and examples of applying our BF taxonomy to describe particular vulnerabilities.

Our Definitions, BF Taxonomy, and Examples

Buffer Overflow (BOF): The software can access through an array a memory location that is outside the array boundaries.



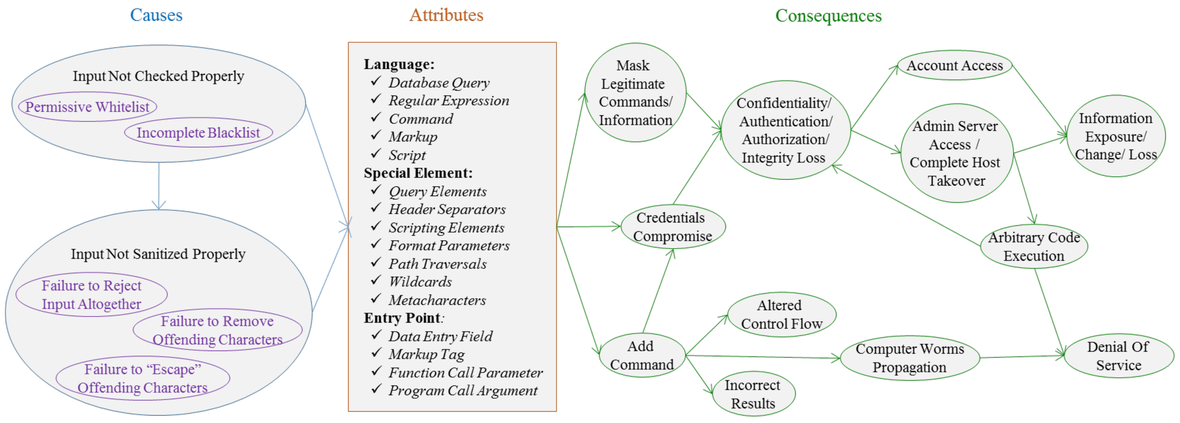
Examples

CVE-2014-0160 (Heartbleed): Input not checked properly leads to too much data, where a huge number of bytes are read from the heap in a continuous reach after the array end, which may be exploited for exposure of information that had not been cleared.

CVE-2015-0235 (Ghost): Incorrect calculation, (specifically missing factor) leads to array too small, where a moderate number of bytes are written to the heap in a discrete reach after the array end, which may be exploited for arbitrary code execution, eventually leading to denial of service.

CVE-2010-1773 (Chrome WebCore): Incorrect calculation, (specifically off by one) leads to a wrong index, where a small number of bytes are read from the heap in a discrete reach before the array start, which may be exploited for information exposure, arbitrary code execution or program crash, leading to denial of service.

Injection (INJ): Due to malicious input with a language-specific special element, the software can assemble a command string that is parsed into an invalid construct.



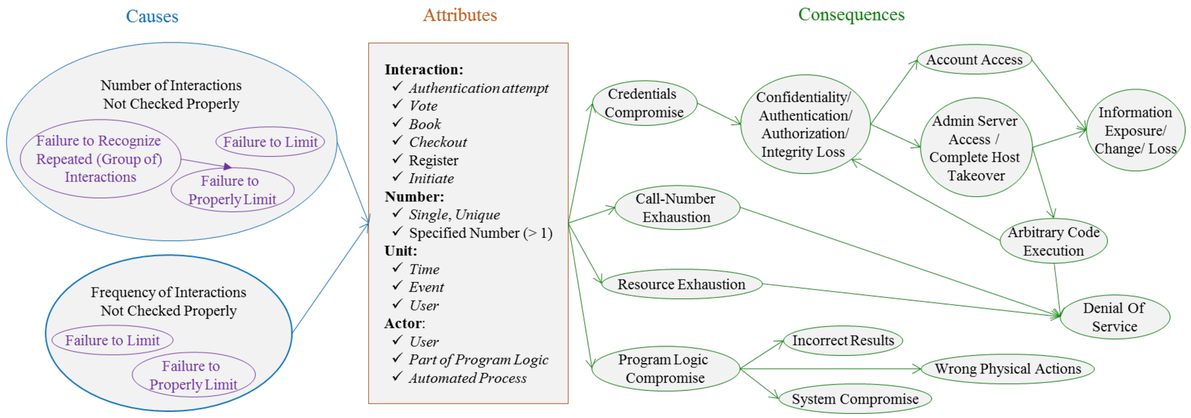
Examples

CVE-2007-3572 (Yoggie Pico): Input not checked properly (specifically incomplete blacklist) allows shell command injection through the "param" function parameter in a CGI script using Shell metacharacters (specifically back ticks `), which may be exploited to add command, leading to arbitrary code execution. Note that adding a command through the function Ping to change the root password enables eventual complete host takeover.

CVE-2008-5817: Input not checked properly or input not sanitized properly allows SQL injection through the "username" and "password" fields in a PHP script using query elements (specifically single quote ' , the word or and equality sign =), which may be exploited to mask legitimate SQL commands, leading to authentication compromise, admin server access and arbitrary SQL code execution.

CVE-2008-5734: Input not sanitized properly allows XSS web script or HTML injection through the IMG element of a generated HTML email, which may be exploited to add commands or for cookie-based authentication credentials compromise, leading to arbitrary code execution.

Interaction Frequency Control (IFC): The software does not properly limit the number of repeating interactions per specified unit.



Examples

CVE-2002-0628: Failure to limit to a specified number the authentication attempts per authentication event by same or different user(s) may be exploited for credentials compromise (username or password) via brute force.

CVE-2002-1876: Failure to recognize repeated interactions that are rapid initiations of message exchange requests from authenticated users, leads to failure to properly limit them to a specified number per specified time interval, which may be exploited for resource exhaustion (consumption of all granted licenses) leading to denial of service.

CVE-2002-1018: Failure to limit the checkouts of a book to a single one per user may be exploited for resource exhaustion leading to denial of service.

Reference for CVE: https://cve.mitre.org