Finding Buffer Overflows

James D. Murray, CISSP C|EH

@jdmurray| www.TechExams.net/blogs/jdmurray

Common Vulnerability Taxonomy

Known Vulnerabilities

Unknown Vulnerabilities

Discovered
Documented
Reproducible
Publically Disclosed

Undiscovered, or discovered but not publically disclosed

Rumsfeldian Vulnerability Taxonomy

Known Unknown Vulnerabilities

Unknown Unknown Vulnerabilities

Suspected or likely to exist, but undiscovered

Not yet realized as potential vulnerabilities

Who Looks for Vulnerabilities?

Known or Unknown

Public, secret, undiscovered, or unimagined

What's your role?

Systems Administrators

- Installed programs know to have BoF vulnerabilities
  - Application vulnerability scanning
- Insufficient patching
- Review the local system event logs for detected buffer overflow events
  - Host-based security software
- Do not ignore program crashes!

Who Looks for Vulnerabilities?

Known or Unknown

Public, secret, undiscovered, or unimagined

What's your role?

Network Administrators

- Review SIEM incident reports for detected buffer overflow events
- Intrusion Detection/Prevention Systems (IDS/IPS/IDP)
- Cyber Threat Intelligence about malicious actors using buffer overflow attacks

Who Looks for Vulnerabilities?

Known or Unknown

Public, secret, undiscovered, or unimagined

What's your role?

Software Developers

- Find and fix BoF issues in legacy code
- Use secure coding practices
- Use secure testing practices

Software Testers and Analysts

- Use through and repeatable testing methods
- Use automated testing tools
- Know how to exploit buffer overflows (for testing)

Who's Responsible for BoF's?

Famous Malware and BoF Vulnerabilities

History of Malware

Input + RAM = BoF

Gotcha
Brain
Lehigh
Stoned

Code Red
Nimda
Sasser
L10n

In The Beginning…

- Morris Worm (1988)
  - First Internet worm and first DDoS attack
  - Exploited BoF vulnerability in fingerd (finger daemon)
  - Overflowed 512-byte buffer to open a reverse shell (/bin/sh)
  - 6000 Unix system infected (10% of Internet)

- "Smashing The Stack For Fun And Profit" by Aleph One (Phrack magazine #49)
  - https://www.win.tue.nl/~aeb/linux/hh/phrack/P49-14

Code Red

- Code Red (2001)
  - Exploited vulnerability in Microsoft IIS 4/5/6 on Windows NT/2000/XP
  - Indexing Service API buffer overflow allowed full access to Windows
  - Worm propagation, Web site defacement, DDoS attacks
  - Microsoft Security Bulletin MS01-033
  - Weaponizing security patches
  - Estimated total cost of recovery from Code Red: $2.1 Billion
  - Variants include Code Red II, Code Blue, and Code Green

Windows' Worm Holes

- Nimda (2001)
  - IIS vulnerabilities, email, writeable Web folders, open network shares
  - Windows 95, 98, ME, NT, 2000, and XP systems, both workstation and server
  - Nimda crashed many of the computers it attempted to infect
- Bolgimo (2003)
  - Microsoft Security Bulletin MS03-026
  - Patched DCOM RPC vulnerability it used to infect Windows NT/2000/XP
  - Removed other Malware (Lovesan/Blaster); Welchia (Nachi) worm did the same
  - Making changes—good or bad—without authorization is unethical

Windows' Worm Holes

- SQL Slammer (2003)
  - Microsoft Database Engine 2000 (MSDE, SQL Server)
  - Gain full control of any Windows system running MSDE—over 75,000 worldwide
  - Conventional A/V could not detect SQL Slammer
  - Microsoft Security Bulletin MS02-039 (July 2002)
- Sasser (2004)
  - Local Security Authority Subsystem Service (LSASS)
  - Shellcode opened a reverse shell to the Internet
  - Attacked Windows 2000/XP systems exposing port 445/tcp to the Internet
  - Reverse engineered Microsoft patch MS04-011

It Doesn't End Here…

- Microsoft Windows
  - Microsoft Windows DCOM RPC Interface Buffer Overrun (MS03-026)
  - Microsoft Windows WebDav Buffer Overflow (MS03-007)
  - Microsoft LSASS Service Overflow (MS04-011)
- UNIX/Linux
  - ISC BIND buffer overflows (L10n worm)
  - RPC statd overflow (Ramen worm)
  - Telnetd (X.C worm)

Buffer Overflows No Longer Exploitable?

Stuxnet (2011)
Duqu (2011)

Flame
(2012)

Cryptolocker
(2013)

Zeus
Banking Trojan
(2007, 2011, 2013)

Heartbleed
(2014)

GNU glibc
(2015, 2016)

Vulnerability Databases and Reports

- Vulnerability databases contain reports of verified, security-related flaws
- Reports are focused on a software program, hardware device, or entire system
- Vulnerability tracking databases
  - Community, business, and government bug tracking databases
  - Security product vendors
  - Vendors whose products have vulnerabilities
  - Security research blogs and forums
  - Exploit databases and repositories

Common Vulnerabilities and Exposures

https://cve.mitre.org/

CVE is a dictionary of publicly known information security vulnerabilities and exposures in publicly released software packages.

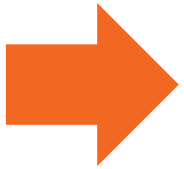
Global and free for public and private use

CVE ID (also name, number, or just "CVE")

CVE-2016-0001

Over 75,000 CVE IDs since 1999

https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=%22buffer+overflow%22



CVE-1999-0030	root privileges via buffer overflow in xlock command on SGI IRIX systems.
CVE-1999-0029	root privileges via buffer overflow in ordist command on SGI IRIX systems.
CVE-1999-0028	root privileges via buffer overflow in login/scheme command on SGI IRIX systems.
CVE-1999-0027	root privileges via buffer overflow in eject command on SGI IRIX systems.
CVE-1999-0026	root privileges via buffer overflow in pset command on SGI IRIX systems.
CVE-1999-0025	root privileges via buffer overflow in df command on SGI IRIX systems.
CVE-1999-0023	Local user gains root privileges via buffer overflow in rdist, via lookup() function.
CVE-1999-0022	Local user gains root privileges via buffer overflow in rdist, via expstr() function.
CVE-1999-0021	Arbitrary command execution via buffer overflow in Count.cgi (wwwcount) cgi-bin program.
CVE-1999-0018	Buffer overflow in statd allows root privileges.
CVE-1999-0009	Inverse query buffer overflow in BIND 4.9 and BIND 8 Releases.
CVE-1999-0008	Buffer overflow in NIS+, in Sun's rpc.nisd program.
CVE-1999-0006	Buffer overflow in POP servers based on BSD/Qualcomm's qpopper allows remote attackers to gain root access using a long PASS command.
CVE-1999-0005	Arbitrary command execution via IMAP buffer overflow in authenticate command.
CVE-1999-0004	MIME buffer overflow in email clients, e.g. Solaris mailtool and Outlook.
CVE-1999-0003	Execute commands as root via buffer overflow in Tooltalk database server (rpc.ttdbserverd).
CVE-1999-0002	Buffer overflow in NFS mountd gives root access to remote attackers, mostly in Linux systems.

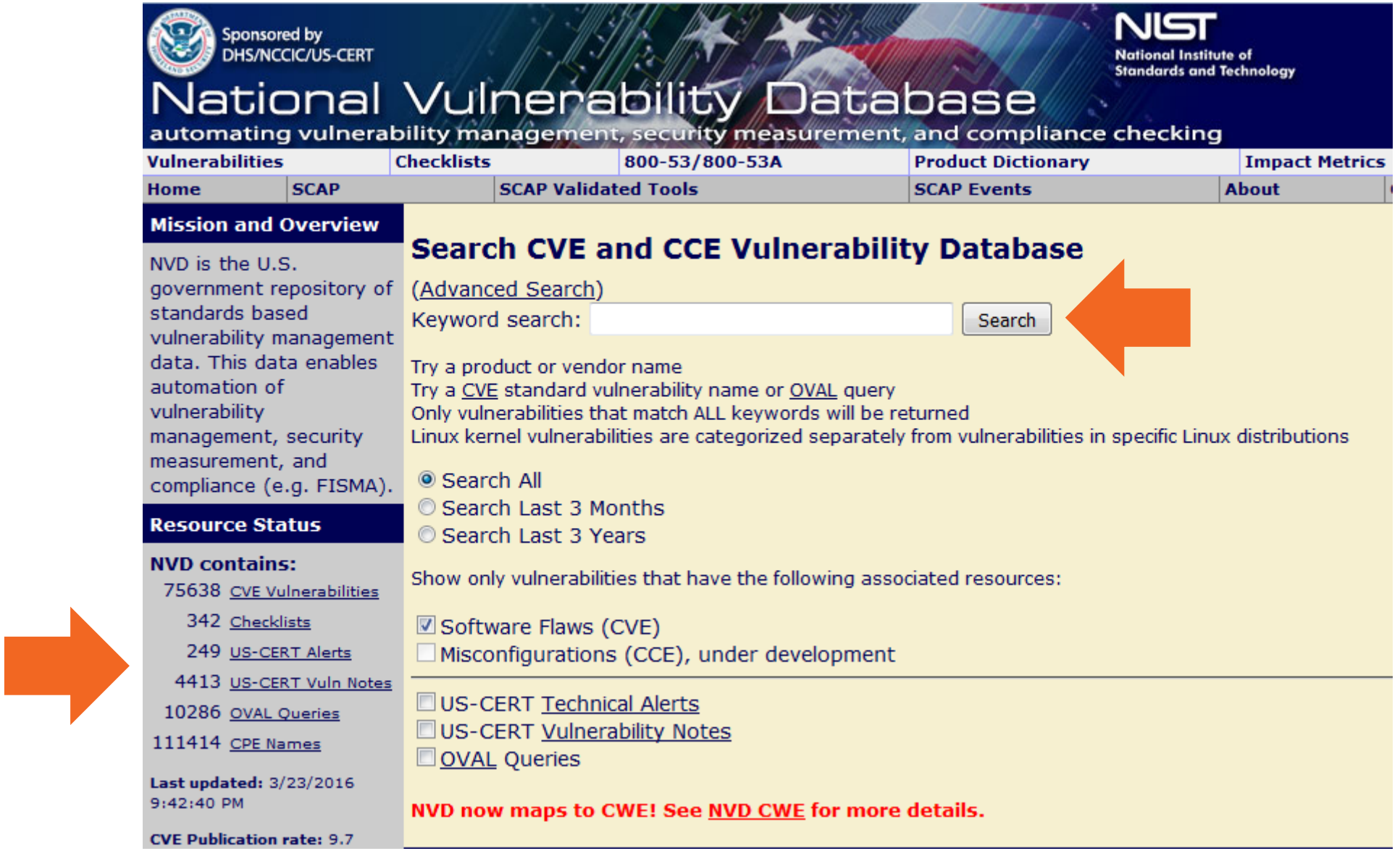
BACK TO TOP

SEARCH CVE USING KEYWORDS: Submit

You can also search by reference using the CVE Reference Maps.

For More Information: cve@mitre.org

https://nvd.nist.gov/home.cfm



Common Weakness Enumeration

http://cwe.mitre.org/

- Public dictionary of software system flaws and vulnerabilities
- "Weakness" includes flaws, errors, vulnerabilities
- Cross-reference vulnerability sources
- Security taxonomies, research, checklists
- Identify, mitigate, and prevent weaknesses
- Provide a measuring stick for security tools

CWE List
Full Dictionary View
Development View
Research View
Fault Pattern View
Reports
Mapping & Navigation
About
Sources
Process
Documents
FAQs
Community
Use & Citations
SwA On-Ramp
Discussion List
Discussion Archives
Contact Us
Scoring
Prioritization
CWSS
CWRAF
CWE/SANS Top 25
Compatibility
Requirements
Coverage Claims Representation
Compatible Products
Make a Declaration
News
Calendar
Free Newsletter

Presentation Filter: --None--

CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer

Improper Restriction of Operations within the Bounds of a Memory Buffer

Weakness ID: 119 *(Weakness Class)* **Status:** Usable

Description

Description Summary

The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.

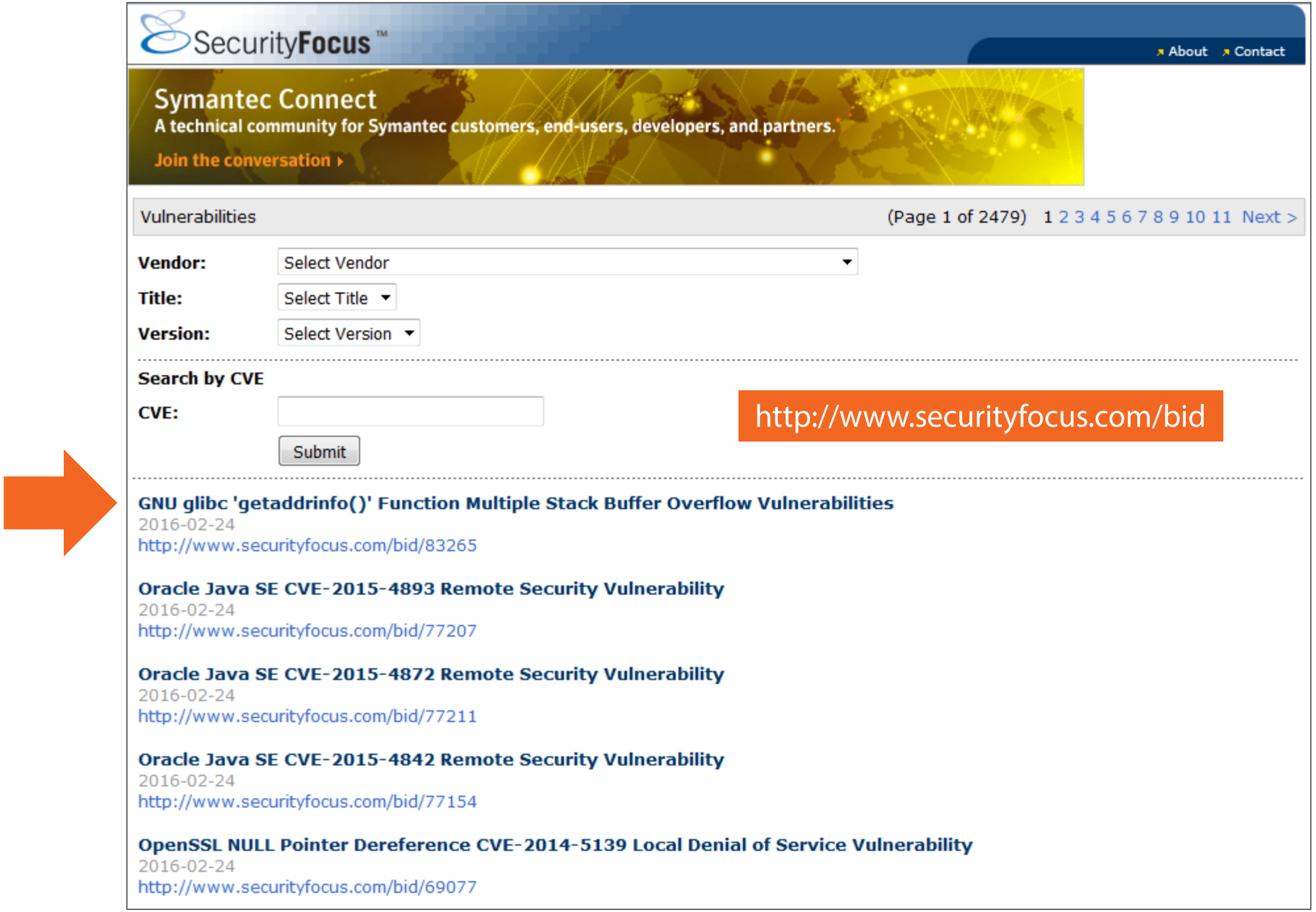
Extended Description

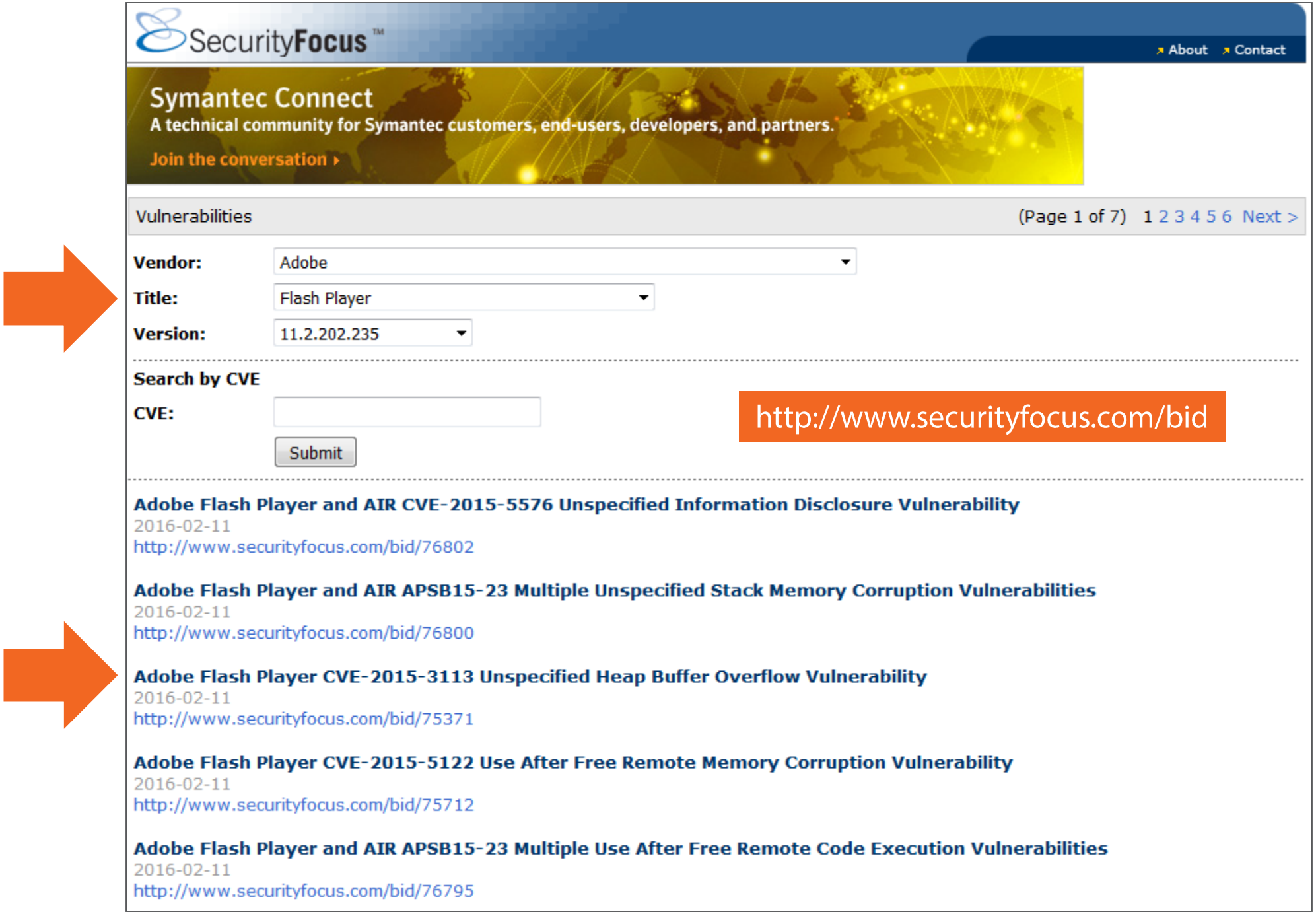
Certain languages allow direct addressing of memory locations and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data.

As a result, an attacker may be able to execute arbitrary code, alter the intended control flow, read sensitive information, or cause the system to crash.

Alternate Terms

Memory Corruption: The generic term "memory corruption" is often used to describe the consequences of writing to memory outside the bounds of a buffer, when the root cause is something other than a sequential copies of excessive data from a fixed starting location (i.e., classic buffer overflows or CWE-120). This may include issues such as incorrect pointer arithmetic, accessing invalid pointers due to incomplete initialization or memory release, etc.


SecurityFocus
About
Contact
Symantec Connect
A technical community for Symantec customers, end-users, developers, and partners.
Join the conversation
Vulnerabilities
(Page 1 of 2479) 1 2 3 4 5 6 7 8 9 10 11 Next >
Vendor:
Select Vendor
Title:
Select Title
Version:
Select Version
Search by CVE
CVE:
Submit
http://www.securityfocus.com/bid
GNU glibc 'getaddrinfo()' Function Multiple Stack Buffer Overflow Vulnerabilities
2016-02-24
http://www.securityfocus.com/bid/83265
Oracle Java SE CVE-2015-4893 Remote Security Vulnerability
2016-02-24
http://www.securityfocus.com/bid/77207
Oracle Java SE CVE-2015-4872 Remote Security Vulnerability
2016-02-24
http://www.securityfocus.com/bid/77211
Oracle Java SE CVE-2015-4842 Remote Security Vulnerability
2016-02-24
http://www.securityfocus.com/bid/77154
OpenSSL NULL Pointer Dereference CVE-2014-5139 Local Denial of Service Vulnerability
2016-02-24
http://www.securityfocus.com/bid/69077


SecurityFocus
About
Contact
Symantec Connect
A technical community for Symantec customers, end-users, developers, and partners.
Join the conversation
Vulnerabilities
(Page 1 of 7) 1 2 3 4 5 6 Next >
Vendor: Adobe
Title: Flash Player
Version: 11.2.202.235
Search by CVE
CVE:
Submit
http://www.securityfocus.com/bid
Adobe Flash Player and AIR CVE-2015-5576 Unspecified Information Disclosure Vulnerability
2016-02-11
http://www.securityfocus.com/bid/76802
Adobe Flash Player and AIR APSB15-23 Multiple Unspecified Stack Memory Corruption Vulnerabilities
2016-02-11
http://www.securityfocus.com/bid/76800
Adobe Flash Player CVE-2015-3113 Unspecified Heap Buffer Overflow Vulnerability
2016-02-11
http://www.securityfocus.com/bid/75371
Adobe Flash Player CVE-2015-5122 Use After Free Remote Memory Corruption Vulnerability
2016-02-11
http://www.securityfocus.com/bid/75712
Adobe Flash Player and AIR APSB15-23 Multiple Use After Free Remote Code Execution Vulnerabilities
2016-02-11
http://www.securityfocus.com/bid/76795

Symantec

Security Response

Our security research centers around the world provide unparalleled analysis of and protection from IT security threats that include malware, security risks, vulnerabilities, and spam.

Overview | Threats | Risks | **Vulnerabilities** | Spam | A-Z

Security Advisory List for Symantec Products | Security Focus Vulnerability Database

Vulnerabilities

Severity	Name	Discovered
▮▮▮▮▯	Microsoft Internet Explorer CVE-2016-0069 Remote Privilege Escalation Vulnerabil...	02/09/2016
▮▮▮▯▯	Microsoft Windows CVE-2016-0041 DLL Loading Multiple Local Privilege Escalation ...	02/09/2016
▮▮▮▮▮	Microsoft Internet Explorer and Edge CVE-2016-0061 Remote Memory Corruption Vuln...	02/09/2016
▮▮▮▮▮	Microsoft Internet Explorer CVE-2016-0063 Remote Memory Corruption Vulnerability	02/09/2016
▮▮▮▮▮	Microsoft Internet Explorer CVE-2016-0060 Remote Memory Corruption Vulnerability	02/09/2016
▮▮▮▮▮	Microsoft Internet Explorer and Edge CVE-2016-0062 Remote Memory Corruption Vuln...	02/09/2016
▮▮▮▮▮	Microsoft Windows Reader CVE-2016-0046 Remote Code Execution Vulnerability	02/09/2016
▮▮▮▮▮	Microsoft Internet Explorer CVE-2016-0072 Remote Memory Corruption Vulnerability	02/09/2016
▮▮▮▯▯	Microsoft ASP.NET Templates Cross Site Request Forgery Vulnerability	02/09/2016
▮▮▮▯▯	Microsoft Active Directory Federation Services CVE-2016-0037 Denial of Service V...	02/09/2016
▮▮▮▮▯	Microsoft Internet Explorer CVE-2016-0068 Remote Privilege Escalation Vulnerabil...	02/09/2016
▮▮▮▮▯	Microsoft Internet Explorer CVE-2016-0071 Remote Memory Corruption Vulnerability	02/09/2016
▮▮▮▮▮	Microsoft Internet Explorer CVE-2016-0067 Remote Memory Corruption Vulnerability	02/09/2016
▮▮▮▮▮	Microsoft Internet Explorer CVE-2016-0064 Remote Memory Corruption Vulnerability	02/09/2016
▮▮▮▯▯	Microsoft Internet Explorer CVE-2016-0059 Information Disclosure Vulnerability	02/09/2016

http://www.symantec.com/security_response/landing/vulnerabilities.jsp

Patch Release Notices

Malware usually exploits publically-disclosed vulnerabilities

Vulnerabilities are first disclosed as patch notices

Check software vendor's Web site and bug tracking system

Full disclosure of details might take a while

mozilla

https://www.mozilla.org/en-US/security/known-vulnerabilities/firefox/

Security Advisories for Firefox

Impact key

CRITICAL	Vulnerability can be used to run attacker code and install software, requiring no user interaction beyond normal browsing.
HIGH	Vulnerability can be used to gather sensitive data from sites in other windows or inject data or code into those sites, requiring no more than normal browsing actions.
MODERATE	Vulnerabilities that would otherwise be High or Critical except they only work in uncommon non-default configurations or require the user to perform complicated and/or unlikely steps.
LOW	Minor security vulnerabilities such as Denial of Service attacks, minor data leaks, or spoofs. (Undetectable spoofs of SSL indicia would have "High" impact because those are generally used to steal sensitive data intended for other sites.)

Fixed in Firefox 44.0.2

Mozilla Security

Security Advisories

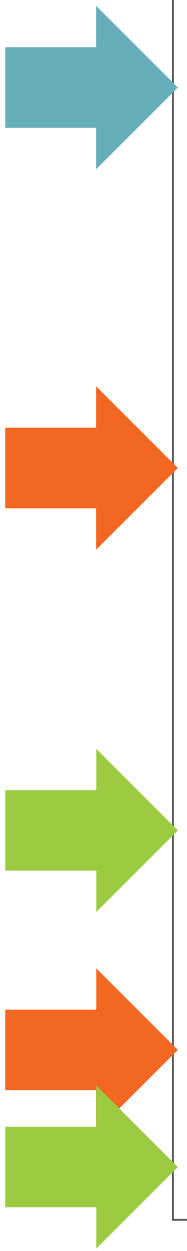
Known Vulnerabilities

Bug Bounty

Firefox Hall Of Fame

Mozilla Web and Services Hall Of Fame

Security Blog



Fixed in Firefox 38

2015-93	Integer overflows in libstagefright while processing MP4 video metadata
2015-58	Mozilla Windows updater can be run outside of application directory
2015-57	Privilege escalation through IPC channel messages
2015-56	Untrusted site hosting trusted page can intercept webchannel responses
2015-55	Buffer overflow and out-of-bounds read while parsing MP4 video metadata
2015-54	Buffer overflow when parsing compressed XML
2015-53	Use-after-free due to Media Decoder Thread creation during shutdown
2015-52	Sensitive URL encoded information written to Android logcat
2015-51	Use-after-free during text processing with vertical text enabled
2015-50	Out-of-bounds read and write in asm.js validation
2015-49	Referrer policy ignored when links opened by middle-click and context menu
2015-48	Buffer overflow with SVG content and CSS
2015-47	Buffer overflow parsing H.264 video with Linux Gstreamer
2015-46	Miscellaneous memory safety hazards (rv:38.0 / rv:31.7)

Bug Tracking Systems

- Defect tracking systems
- Problem reporting systems
- Problem reports, feature requests, and testing results
- Build software patches, history of code and project changes
- Bug tracking systems are the best source of security information

Vulnerability Exploit Databases

- Information on how to exploit vulnerabilities
  - Exploit tools
  - Exploit payloads (shellcode)
  - Tips & tricks (a.k.a, “hacks”)
  - Papers, documents, blogs, discussion forums
- Exploit Websites
  - https://www.exploit-db.com/
  - http://www.rapid7.com/db/
  - http://0day.today/
  - https://cxsecurity.com/exploit/

Programming Standards and Guidelines

Programmer, software developer, coder, software system engineer

Programming standards and guidelines

Design, implementation, and testing methods

Robust, secure, bug-free systems

Find and fix existing bugs and security issues

Use Which for What?

Standards
Guidelines
Recommendations
Best Practices

Shared rules, methods, patterns, and language

Instructions how to implement a standard or a policy

Like a standard, but not official

Proven to the best solution to a problem

?

Secure Code Development Standards and Practices

- Unsafe programming languages
- Safe(r) programming languages
- Unsafe functions, libraries, APIs
- Safe(r) functions, libraries, APIs
- Input validation
- Boundary checking

- strcpy
- strcpy_s
- strcat
- strcat_s
- gets
- printf

- C/C++
- Assembly languages

- .NET Framework languages
- Java
- Perl
- Python (interpreted)
- Ruby

- sprintf_s

- Is "x@y.com' OR 1=1 --" valid input?
- Is the value 5 between 0 and 9?

SEI CERT Coding Standards

SEI coding standard

- Rules for writing secure code
- Recommendations for improving software security

https://www.securecoding.cert.org/confluence/display/seccode/

SEI CERT Top Ten Coding Practices

1. Validate input
2. Heed compiler warnings
3. Architect and design for security policies
4. Keep it simple
5. Default deny
6. Adhere to the principle of least privilege
7. Sanitize data sent to other systems
8. Practice defense in depth
9. Use effective quality assurance techniques
10. Adopt a secure coding standard

Open Web Application Security Project

https://www.owasp.org/

Improving software security by publishing tools and information aiding in the management of software security risks

Manage risk, not just vulnerabilities

Vulnerability assessment tools

Learning Web application penetration testing

OWASP tools and information are FOSS

All are welcome at OWASP!

OWASP Top 10 Web App Security Flaws

1. A1 Injection
2. A2 Broken Authentication and Session Management
3. A3 Cross-Site Scripting (XSS)
4. A4 Insecure Direct Object References
5. A5 Security Misconfiguration
6. A6 Sensitive Data Exposure
7. A7 Missing Function Level Access Control
8. A8 Cross-Site Request Forgery (CSRF)
9. A9 Using Components with Known Vulnerabilities
10. A10 Unvalidated Redirects and Forwards

Microsoft Security Development Lifecycle

https://www.microsoft.com/sdl/
https://www.microsoft.com/twc/

Reduce both security vulnerabilities and development costs

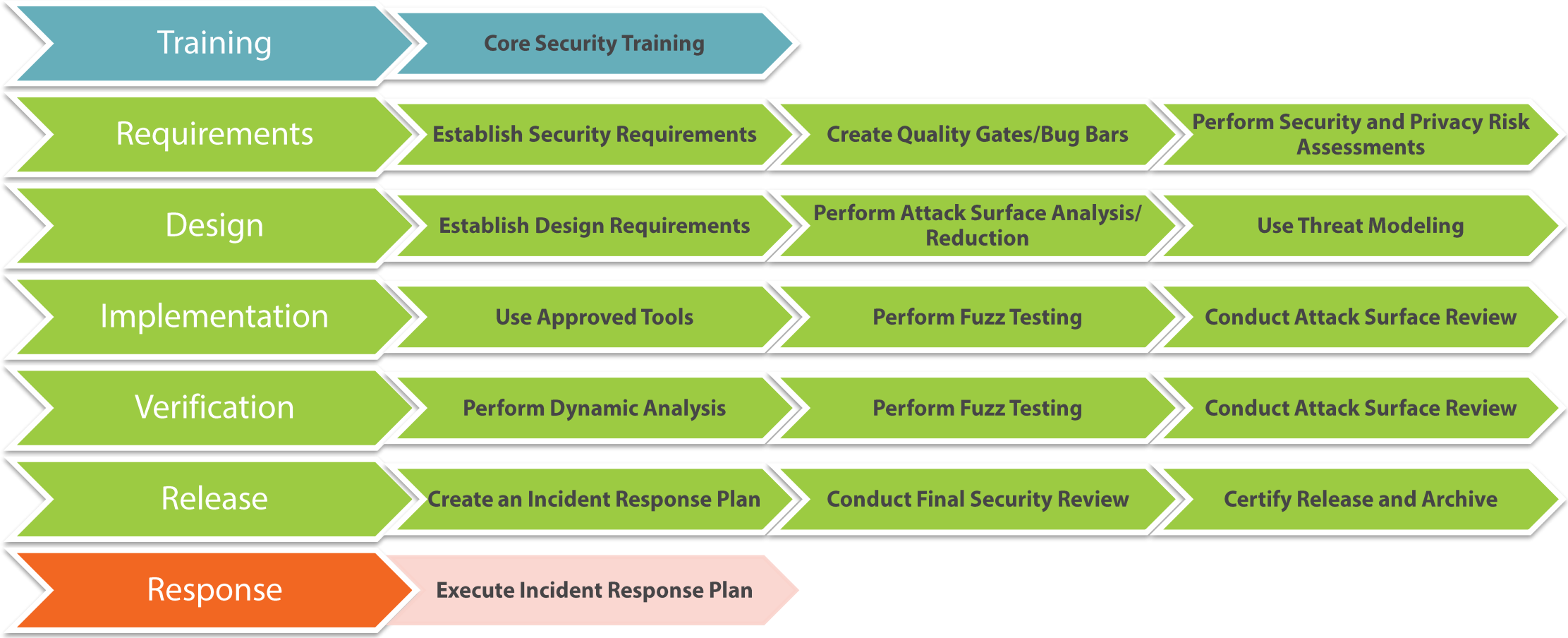
Microsoft Trustworthy Computing Initiative

Security, Privacy, Reliability, and Business Integrity

SDL released to public in 2008

Following the SDL standard is recommended for all organizations development software for the Microsoft Windows environments.

Microsoft Security Development Lifecycle



Who's Responsible for Secure Code?

- Software Developers
  - Use secure coding practices to design and implement code
  - Range checking, input validation, checking return values
  - Do not use unsafe calls in code libraries.
  - Do not use 3rd-party libraries that may be unsafe.
  - Use compiler options that detect and handle overflow conditions.
  - Use programming languages that have internal buffer management
  - Review code for potential buffer mismanagement problems
    - Use automated code review tools for obvious implementation problems
    - Use human review for design flaws that can be exploited

Who's Responsible for BoF's?

- Software Testers
  - No code to look at (Black Box testing)
  - Test all human and programmatic (API) inputs
  - Range checking, input validation, checking return values
  - Use a wide variety of data types and lengths for input testing
  - Overflow conditions (memory faults) may occur without data input
  - Develop procedure for reliably reproducing the error
  - Talk with the software developers

Static Code Analysis

Looking for problems in a program's source code

Automatic source code analysis

Manual source code analysis

White Box testing
or
Clear Box analysis

What is Source Code?

```
int main()
{
    int i;
    int datasize = 200000000;
    char *p = storeData("Test String", datasize);
    for (i = 0; i < datasize; i++)
        putchar(*(p+i));
    putchar('\n');
}

char *storeData(char *data, int datasize) {
    int i;
    char *p = (char *)malloc(datasize);
    if (p != (char *)NULL)
        memcpy((char *)p, data, datasize);
    return p;
}
```

Instructions describing what a computer program can potentially do

Static analysis looks at source code files, not programs running in memory

Input data controls what a program does

What data will the program receive as input?

Coding Problems That Create BoF Conditions

Misuse and mismanagement of program memory

Over writing data, Data corruption, Bad memory addressing

Find, fix, and prevent BoF coding mistakes

The C language (C++ and assembly language too)

Function misuse
Input Validation
Boundary checking

Spend some time looking at and learning to code

Functions, Methods, Procedures, APIs

- Functions contain all code in a program
- Functions are internal to a program
- Functions are typically only accessible from within their program
- Remote Procedure Calls (RPC) can be accessed from outside of a program
- Application Programming Interface (API) exposes functions to other programs
- APIs can be local to the computer or exposed to a network

Function Safety

- Improper coding and use of function can create buffer overflow conditions
- Three categories of function safety:
  1. Functions easy to use safely - safe
  2. Functions easy to use unsafely - hazardous
  3. Functions impossible to use safely – dangerous

- Just throw away all that ancient, legacy code and rewrite it!

Safe and Unsafe Functions

Unsafe(Banned) Functions

- gets
- strcpy, strncpy
- strcat, strncat
- sprintf
- scanf, sscanf
- memcpy

Safe Function Equivalents

- gets_s
- strcpy_s, strncpy_s
- strcat_s, strncat_s
- sprintf_s
- sscanf_s
- memcpy_s

Safe and Unsafe Functions

Unsafe(Banned) Functions

- strcpy, strcpyA, strcpyW, wcscpy,
- _tcscpy, _mbscpy, StrCpy, StrCpyA,
- StrCpyW, lstrcpy, lstrcpyA, lstrcpyW,
- _tccpy, _mbccpy, _ftcscpy, strncpy,
- wcsncpy, _tcsncpy, _mbsncpy,
- _mbsnbcpy, StrCpyN, StrCpyNA,
- StrCpyNW, StrNCpy, strcpynA, StrNCpyA,
- StrNCpyW, lstrcpyn, lstrcpynA, lstrcpynW

Safe Function Equivalents

- strcpy_s

Unsafe to Use Ever!

```
char in_buffer[40];
printf("Password: ");
gets(in_buffer);
```

```
Password: ****************************************
Segmentation fault
```

But Are Safe Functions Really Safer?

Yes, because they use safe coding

Boundary checking, input/ parameter validation

Static code analysis finds calls to unsafe functions

Compilers warn of and even replace unsafe functions

Boundary Checking

- Code and data has boundaries
- Stay within the lines…
- …or risk overflowing
- Don't forget boundary checking!

- Illegal array index
- Incorrect data assignment
- Sanity-check index variable value

```
char buffer[10]; // index values 0 - 9
buffer[5] = "Good";
buffer[10] = "Bad";
buffer[-1] = "Bad";
buffer[5] = (int)"H";
buffer[1] = 701;
int j = 42;
buffer[j] = j;  // index values 0 - 41
```

Input Validation

- Input validation, input checking, input filtering
- Parameter validation
- Data is not checked before it is passed to functions
- Function do not check data before they use it
- Functions should always reject invalid input data
- Input sanitization may be able to correct invalid data
- johnsmith SANATIZED JOHNSMITH

Rules of Input Validation

- Rules of Input validation
  - The data size acceptable?
  - Is the data type acceptable?
  - Is the data content acceptable?
  - Is the data format acceptable?
  - Does the data exist at the location specified?
- Programs and functions should never use input data without first checking it for validity.
- Program and function output should be checked for validity as well.

Parameter Validation

```
#include <stdio.h>
int main(int argc, char* argv[])
{
    printf("%s", argv[2]);
}
```

argv[2] is passed to printf without parameter validation

- myprogram arg1 arg2 arg3
- If argv[2] exists is not checked
- If argv[2] contains data is not checked

Parameter Validation

```
#include <stdio.h>
int main(int argc, char* argv[])
{
    if (argc > 1)
        if (argv[2] != (char*)NULL)
            if (strlen(argv[2]) < 80)
                printf("%s", argv[2]);
}
```

Check if argv[2] might exist

Check if argv[2] contains data

Check the length of the data

Format String Vulnerability

```
char fmtarg[] = “Name: %s\n”;
printf(fmtarg, username);


printf(“Name: %s\n”, username);
```

Format String Vulnerability

```
void printUsernameTwice(char* fmtarg, char* username)
{
    printf(fmtarg, username);

    printf("Name: %s\n", username);
}
```

Binary Code Analysis

Code security review, but no source code?

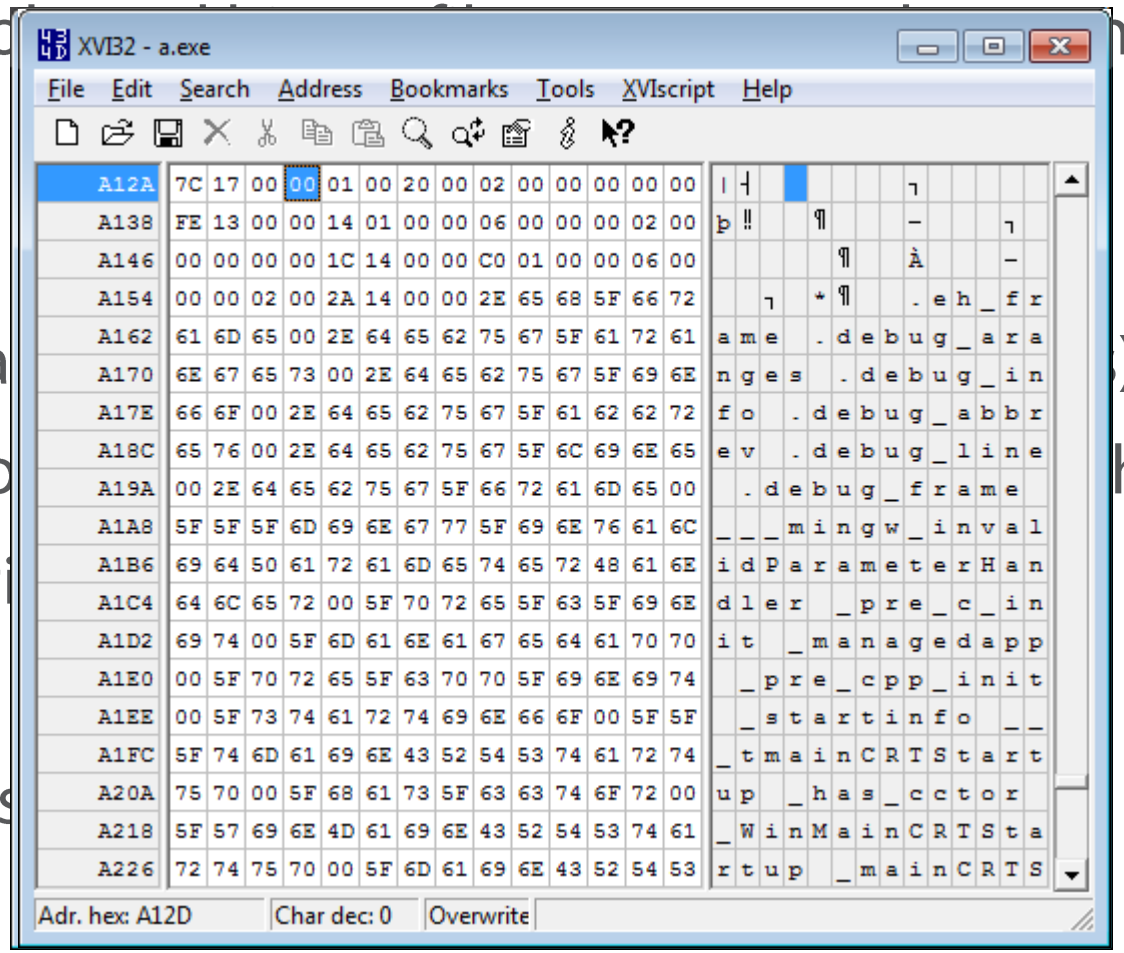
Analyze the binary code in the program's executable file

Binary code, a.k.a., object code & machine code

Executable files
library files
module files

Binary Code Analysis

- Binary analysis too
- String data
- Functions called
- Memory mismana
- Blacklisted functio
- Problematic functi
- Binary library and
- Binary code analys



Binary Code vs. ByteCodes

- Oracle Java and Microsoft .NET Framework
- Compiled Programming Languages
  - Source Code → Machine Code
- Interpreted Programming Languages
  - Source Code → Bytecodes
- Bytecodes are read by a language runtime virtual machine
- .NET Source Code → .NET Bytecodes → .NET VM → CPU
- Java Source Code → Java Bytecodes → Java VM → CPU

Static Code Analysis Tools

- Free and Open Source Software
  - FxCop .NET (Microsoft Visual Studio)
  - Clang Static Analyzer (C, C++, Objective-C)
  - Infer (Java, C, Objective-C)
  - FindBugs plugin (Java)
  - cppcheck (C++)
  - Binary Analysis Tool
- Commercial Tools

Static or Binary/Bytecode or Both?

- Static analysis of source code and binary code and bytecodes
- Which one(s) to use?
- Use them all!
- Having the program's source code is best.
- Binary files have a lot of extra code not found in the source code.
  - Statically-linked modules and dynamically-linked libraries
- Bytecode files are worth analyzing, even if you have their source code.

Is this Reverse Engineering?

Deconstruction to discover form, content, purpose

Electrical
Mechanical
Electronic
Intellectual

Reverse Engineering can be illegal

Digital Millennium Copyright Act (DCMA)

Illegal to break copy protection mechanisms

…but you need permission from the copyright holder

Should I Reverse Engineer?

DMCA protects software assets from copying, but allows for security research

Is Malware also protected by the DMCA?

Electronic Frontier Foundation

https://www.eff.org/issues/dmca

Automated Code Analysis

- Manual drudgery or…
- …automation!
- Automated code vulnerability scanning
  - Locating possible implementation security issues
  - Identifying non-security-specific issues (e.g., poor memory management)
  - Verifying presence of error checking/correction code
  - Checking conformance to secure coding standards

Automated Testing Tools

Static and Dynamic

Static code is not running as a program

Dynamic code is running in a testing environment (debugger)

All testing tools have some degree of automation

Static Code Analysis Tools

- Source code compilers
  - Gnu Compiler Collection (https://gcc.gnu.org/)
  - Clang (http://clang-analyzer.llvm.org/)
- Integrated Development Environments (IDE)
  - Analysis tools integrated into the IDE and run automatically
  - Microsoft Visual Studio IDE
    - PREFast – C/C++static source code analsysic
    - FxCop - .NET assembly CIL (bytecode) analysis
- Stand-alone static analysis tools

Dynamic Code Analysis

- Dynamic analysis is performed on code that is running as a program.
- White Box testing
  - Code is running in a debugger
  - Code is tested from the inside
  - Integrated Development Environments (IDE)
- Black Box testing
  - Code is running as a process (on a test box or production machine)
  - Code is tested from the outside
  - Vulnerability analysis and penetration testing

Buffer Overflow Testing Tools

- Testing tools that exploit buffer overflow vulneraries in running code
  - Penetration testing or Black Box testing tools
- OllyDbg - http://www.ollydbg.de
  - Assembler-level analyzer/debugger for Microsoft Windows binaries
  - Supports automation and plug-ins
  - Shareware (free!) and very popular
- IDA (Interactive Disassembler) – https://www.hex-rays.com/products/ida/
  - Binary code disassembler and debugger for Windows Linux, Mac OS X
  - Remote and scriptable debugging
  - Commercial (IDA Professional) and free (IDS Starter) releases

Buffer Overflow Testing Tools

- Spike - http://www.immunitysec.com/downloads/
  - A toolkit for creating fuzzing testing tools
  - Sends malformed input data across network connections
  - Used with tools like OllyDbg, Wireshark, Nmap, netcat
  - Distributed with Kali Linux - https://www.kali.org/
- Metasploit - http://www.metasploit.com/
  - Professional network and application penetration testing tool
  - Craft and test exploits against known and unknown software vulnerabilities
  - Metasploit Pro (commercial) and Community (free)

Input Testing

Test software using both legal and illegal input values

Range
1..10

Set
y, n, yes, no

Length
8..20 chars

Content
[a..z][A..Z][0..9][!#@]

Format
NNN-NN-NNNN

Input Testing Values

- Use a minimal set of testing values to save time
- Legal input: 1 .. 10
- Legal test values: 1, 5, 10
- Illegal input: All other possible input values
- Illegal test values: -9999, -100, -10, -1, 11, 100, 9999

Fuzz Testing

- Fuzzing
- Use a very large set of random testing values
- Fuzz is randomly-generated numbers, strings, or binary data
- Varies in length, content, and format
- Fuzz input values: 
- http://resources.infosecinstitute.com/intro-to-fuzzing/

Reporting Buffer Overflows

Static Analysis

Binary/Bytecode Analysis

Dynamic Analysis

Release notes, Web forum, blog

Bug tracking system

Vulnerability & exploit databases

That's Not a Bug, It's a Feature!

- What you found is not a problem, it's really a…
  - Misunderstanding form or function
  - False Positive – looks like a problem, but it's not
  - Technically a problem, but not one related to security
- Report to developer's public forum or other programming Website
- Email developer's for private communications

Has It Already Been Fixed?

- Reproduce the issue in the latest release of the software.
- If not present, check for a mention of a fix in the release notes.
- If no mention of fix, maybe report the issue anyway.
- Report a problem in an older release that's fixed in a newer release?
- The problem may have been accidently fixed.
- The software maintainers are unaware of the security issue in older releases.
- Undiscovered bugs in older program releases are still potential problems.

Yes, I Found a Zero-day! Now What?

Report it and forget it

Private and responsible disclosure

Do not disclose publically—at first

Give owners a chance to patch and make public

Reporting a (Possible) Zero-day

- De facto rules for disclosure:
  - Privately contact the program's owner or maintainers
  - Use the reporting method specified by the program owner
  - Do not make a public problem report unless asked to do so
  - Use the reporting format specified by the program owner
  - Provide instructions for duplicating the issue
  - Give the program owner time to respond to your report
  - Work with the program's developer's if asked
  - Report to a reputable White Hat vulnerability organization

Vulnerability Disclosure? Is This Legal?

- Ethical and legal consequences of vulnerability disclosure
- Can I get into trouble?
- Full Disclosure
  - Systems should be fully available for examination by anyone
  - Information about security vulnerabilities should be:
    - published as soon as possible
    - made available to everyone

Full Disclosure

- Encourage user and security researchers to look for exploitable vulnerabilities
- Owners, fix your insecure systems!
- Spread awareness of vulnerabilities to prevent them
- The Dark Side of Full Disclosure
  - Black Hats will become aware of vulnerabilities and craft exploits for them
  - Civil actions for damages against vulnerability disclosers?

- Private disclose to program owners
- Ethical behavior is the best defense

Bug Bounty? Cha-ching!

- Bug Bounty Program
- Money paid for finding security problems in software
  - and disclosing them to the software's owners.
- Many software-producing businesses have Bug Bounty Programs
  - or will have them soon.
- Gain reputation as a successful White Hat security researcher.
- Help keep computing systems safe to use.

The Dark Side of Cash for Bugs

- Bug Brokers (a.k.a., Vulnerability Brokers)
- Cash for Zero-day Vulnerabilities
- Bug Brokers sell Zero-day Exploits too—usually for lots of money
- Ethically Gray Hat to Black Hat
- Bug Brokers usually pay higher then Bug Bounty Programs
  - but want exploitable vulnerabilities that can be attacked

- My advice: Stay White Hat

Summary

- ✓Known and unknown vulnerabilities
- ✓Who finds and fixes vulnerabilities?
- ✓Why do buffer overflows exist in our code?
- ✓Malware and Worm Holes
- ✓Vulnerability reports, dictionaries, databases
- ✓Malware exploit information

Summary

- ✓Coding Standards and Best Practices
- ✓Used to write new code and fix old code
- ✓SEI, OWASP, Microsoft
- ✓Secure Software Development Lifecycle
- ✓Risk Management
- ✓Boundary checking, input validation, and unsafe functions

Summary

- ✓Static source code analysis
- ✓Binary and Bytecode file analysis
- ✓Dynamic code and program analysis
- ✓Automate what you can!
- ✓Reverse engineering (be aware!)
- ✓Reporting bugs ethically and for profit

"All input is evil until proven otherwise."

— Michael Howard, "Writing Secure Code"

Next Up:

Mitigating Buffer Overflows