xLua Scan Report

Project Name	xLua
Scan Start	Friday, June 21, 2024 11:53:14 AM
Preset	Checkmarx Default
Scan Time	00h:08m:03s
Lines Of Code Scanned	15289
Files Scanned	11
Report Creation Time	Friday, June 21, 2024 12:03:23 PM
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012
Team	CxServer
Checkmarx Version	8.7.0
Scan Type	Full
Source Origin	LocalPath
Density	5/1000 (Vulnerabilities/LOC)
Visibility	Public

Filter Settings

Severity

Included: High, Medium, Low, Information

Excluded: None

Result State

Included: Confirmed, Not Exploitable, To Verify, Urgent, Proposed Not Exploitable

Excluded: None

Assigned to

Included: All

Categories

Included:

Uncategorized	All
Custom	All
PCI DSS v3.2	All
OWASP Top 10 2013	All
FISMA 2014	All
NIST SP 800-53	All
OWASP Top 10 2017	All
OWASP Mobile Top 10 2016	All

Excluded:

Uncategorized	None
Custom	None
PCI DSS v3.2	None
OWASP Top 10 2013	None
FISMA 2014	None

NIST SP 800-53	None
OWASP Top 10 2017	None
OWASP Mobile Top 10 2016	None

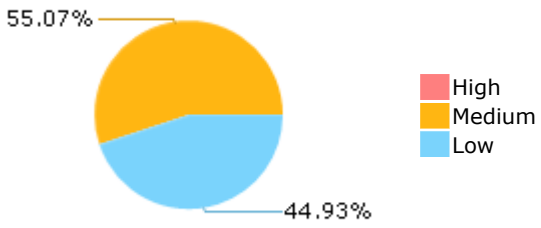
Results Limit

Results limit per query was set to 50

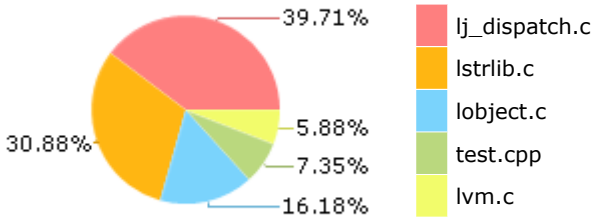
Selected Queries

Selected queries are listed in Result Summary

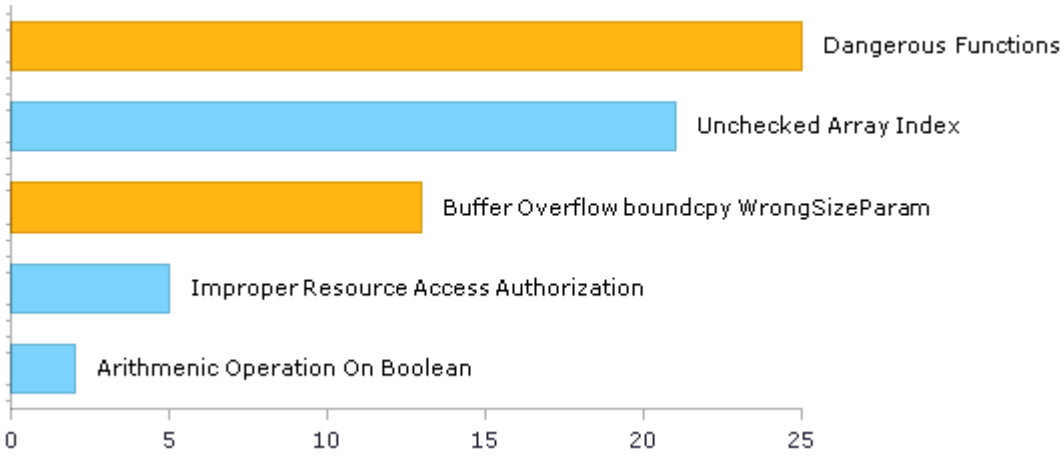
Result Summary



Most Vulnerable Files



Top 5 Vulnerabilities



Scan Summary - OWASP Top 10 2017

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2017

Category	Threat Agent	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection	App. Specific	EASY	COMMON	EASY	SEVERE	App. Specific	14	14
A2-Broken Authentication	App. Specific	EASY	COMMON	AVERAGE	SEVERE	App. Specific	5	5
A3-Sensitive Data Exposure	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App. Specific	0	0
A4-XML External Entities (XXE)	App. Specific	AVERAGE	COMMON	EASY	SEVERE	App. Specific	0	0
A5-Broken Access Control*	App. Specific	AVERAGE	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A6-Security Misconfiguration	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A7-Cross-Site Scripting (XSS)	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A8-Insecure Deserialization	App. Specific	DIFFICULT	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A9-Using Components with Known Vulnerabilities*	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	MODERATE	App. Specific	25	25
A10-Insufficient Logging & Monitoring	App. Specific	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	App. Specific	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - OWASP Top 10 2013

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2013

Category	Threat Agent	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	AVERAGE	SEVERE	ALL DATA	0	0
A2-Broken Authentication and Session Management	EXTERNAL, INTERNAL USERS	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	AFFECTED DATA AND FUNCTIONS	0	0
A3-Cross-Site Scripting (XSS)	EXTERNAL, INTERNAL, ADMIN USERS	AVERAGE	VERY WIDESPREAD	EASY	MODERATE	AFFECTED DATA AND SYSTEM	0	0
A4-Insecure Direct Object References	SYSTEM USERS	EASY	COMMON	EASY	MODERATE	EXPOSED DATA	0	0
A5-Security Misconfiguration	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	EASY	MODERATE	ALL DATA AND SYSTEM	0	0
A6-Sensitive Data Exposure	EXTERNAL, INTERNAL, ADMIN USERS, USERS BROWSERS	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	EXPOSED DATA	0	0
A7-Missing Function Level Access Control*	EXTERNAL, INTERNAL USERS	EASY	COMMON	AVERAGE	MODERATE	EXPOSED DATA AND FUNCTIONS	0	0
A8-Cross-Site Request Forgery (CSRF)	USERS BROWSERS	AVERAGE	COMMON	EASY	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A9-Using Components with Known Vulnerabilities*	EXTERNAL USERS, AUTOMATED TOOLS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	25	25
A10-Unvalidated Redirects and Forwards	USERS BROWSERS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0

* Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - PCI DSS v3.2

Category	Issues Found	Best Fix Locations
PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection	0	0
PCI DSS (3.2) - 6.5.2 - Buffer overflows	13	13
PCI DSS (3.2) - 6.5.3 - Insecure cryptographic storage	0	0
PCI DSS (3.2) - 6.5.4 - Insecure communications	0	0
PCI DSS (3.2) - 6.5.5 - Improper error handling*	0	0
PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)	0	0
PCI DSS (3.2) - 6.5.8 - Improper access control	0	0
PCI DSS (3.2) - 6.5.9 - Cross-site request forgery	0	0
PCI DSS (3.2) - 6.5.10 - Broken authentication and session management	0	0

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - FISMA 2014

Category	Description	Issues Found	Best Fix Locations
Access Control	Organizations must limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems) and to the types of transactions and functions that authorized users are permitted to exercise.	0	0
Audit And Accountability*	Organizations must: (i) create, protect, and retain information system audit records to the extent needed to enable the monitoring, analysis, investigation, and reporting of unlawful, unauthorized, or inappropriate information system activity; and (ii) ensure that the actions of individual information system users can be uniquely traced to those users so they can be held accountable for their actions.	2	2
Configuration Management	Organizations must: (i) establish and maintain baseline configurations and inventories of organizational information systems (including hardware, software, firmware, and documentation) throughout the respective system development life cycles; and (ii) establish and enforce security configuration settings for information technology products employed in organizational information systems.	0	0
Identification And Authentication*	Organizations must identify information system users, processes acting on behalf of users, or devices and authenticate (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing access to organizational information systems.	5	5
Media Protection	Organizations must: (i) protect information system media, both paper and digital; (ii) limit access to information on information system media to authorized users; and (iii) sanitize or destroy information system media before disposal or release for reuse.	0	0
System And Communications Protection	Organizations must: (i) monitor, control, and protect organizational communications (i.e., information transmitted or received by organizational information systems) at the external boundaries and key internal boundaries of the information systems; and (ii) employ architectural designs, software development techniques, and systems engineering principles that promote effective information security within organizational information systems.	0	0
System And Information Integrity	Organizations must: (i) identify, report, and correct information and information system flaws in a timely manner; (ii) provide protection from malicious code at appropriate locations within organizational information systems; and (iii) monitor information system security alerts and advisories and take appropriate actions in response.	0	0

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - NIST SP 800-53

Category	Issues Found	Best Fix Locations
AC-12 Session Termination (P2)	0	0
AC-3 Access Enforcement (P1)	5	5
AC-4 Information Flow Enforcement (P1)	0	0
AC-6 Least Privilege (P1)	0	0
AU-9 Protection of Audit Information (P1)	0	0
CM-6 Configuration Settings (P2)	0	0
IA-5 Authenticator Management (P1)	0	0
IA-6 Authenticator Feedback (P2)	0	0
IA-8 Identification and Authentication (Non-Organizational Users) (P1)	0	0
SC-12 Cryptographic Key Establishment and Management (P1)	0	0
SC-13 Cryptographic Protection (P1)	0	0
SC-17 Public Key Infrastructure Certificates (P1)	0	0
SC-18 Mobile Code (P2)	0	0
SC-23 Session Authenticity (P1)*	0	0
SC-28 Protection of Information at Rest (P1)	0	0
SC-4 Information in Shared Resources (P1)	0	0
SC-5 Denial of Service Protection (P1)*	3	3
SC-8 Transmission Confidentiality and Integrity (P1)	0	0
SI-10 Information Input Validation (P1)*	21	21
SI-11 Error Handling (P2)*	0	0
SI-15 Information Output Filtering (P0)	0	0
SI-16 Memory Protection (P1)	0	0

***** Project scan results do not include all relevant queries. Presets and\or Filters should be changed to include all relevant standard queries.

Scan Summary - OWASP Mobile Top 10 2016

Category	Description	Issues Found	Best Fix Locations
M1-Improper Platform Usage	This category covers misuse of a platform feature or failure to use platform security controls. It might include Android intents, platform permissions, misuse of TouchID, the Keychain, or some other security control that is part of the mobile operating system. There are several ways that mobile apps can experience this risk.	0	0
M2-Insecure Data Storage	This category covers insecure data storage and unintended data leakage.	0	0
M3-Insecure Communication	This category covers poor handshaking, incorrect SSL versions, weak negotiation, cleartext communication of sensitive assets, etc.	0	0
M4-Insecure Authentication	This category captures notions of authenticating the end user or bad session management. This can include: -Failing to identify the user at all when that should be required -Failure to maintain the user's identity when it is required -Weaknesses in session management	0	0
M5-Insufficient Cryptography	The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was attempted, but it wasnt done correctly.	0	0
M6-Insecure Authorization	This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.). If the app does not authenticate users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure.	0	0
M7-Client Code Quality	This category is the catch-all for code-level implementation problems in the mobile client. That's distinct from server-side coding mistakes. This would capture things like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device.	0	0
M8-Code Tampering	This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or	0	0

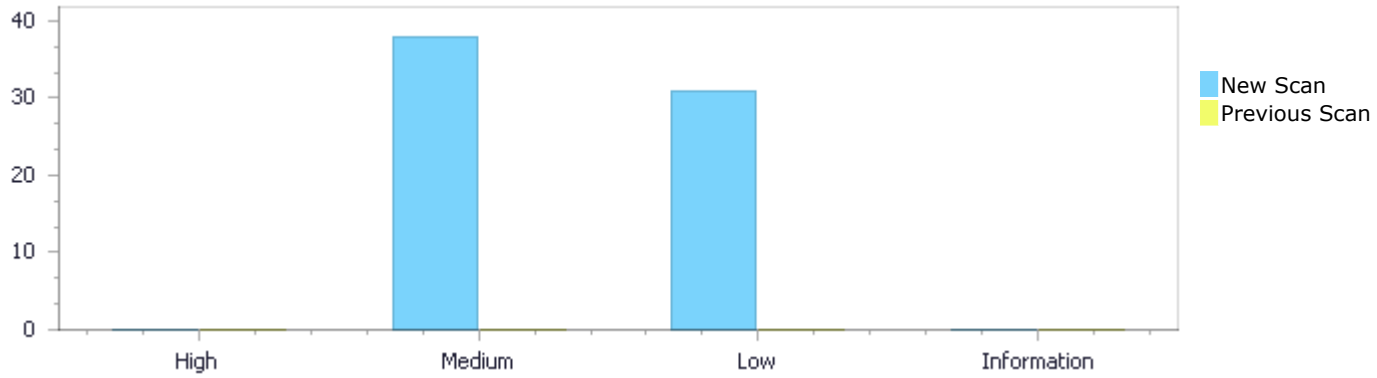
	modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.		
M9-Reverse Engineering	This category includes analysis of the final core binary to determine its source code, libraries, algorithms, and other assets. Software such as IDA Pro, Hopper, otool, and other binary inspection tools give the attacker insight into the inner workings of the application. This may be used to exploit other nascent vulnerabilities in the application, as well as revealing information about back end servers, cryptographic constants and ciphers, and intellectual property.	0	0
M10-Extraneous Functionality	Often, developers include hidden backdoor functionality or other internal development security controls that are not intended to be released into a production environment. For example, a developer may accidentally include a password as a comment in a hybrid app. Another example includes disabling of 2-factor authentication during testing.	0	0

Scan Summary - Custom

Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

Results Distribution By Status First scan of the project

	High	Medium	Low	Information	Total
New Issues	0	38	31	0	69
Recurrent Issues	0	0	0	0	0
Total	0	38	31	0	69
Fixed Issues	0	0	0	0	0



Results Distribution By State

	High	Medium	Low	Information	Total
Confirmed	0	0	0	0	0
Not Exploitable	0	0	0	0	0
To Verify	0	38	31	0	69
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	0	38	31	0	69

Result Summary

Vulnerability Type	Occurrences	Severity
Dangerous Functions	25	Medium
Buffer Overflow boundcpy WrongSizeParam	13	Medium
Unchecked Array Index	21	Low
Improper Resource Access Authorization	5	Low
Arithmenic Operation On Boolean	2	Low

Use of Sizeof On a Pointer Type	2	Low
NULL Pointer Dereference	1	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
xLua/lstrlib.c	18
xLua/lobject.c	10
xLua/lj_dispatch.c	6
xLua/lvm.c	4

Scan Results Details

Dangerous Functions

Query Path:
CPP\Cx\CPP Medium Threat\Dangerous Functions Version:1

Categories

OWASP Top 10 2013: A9-Using Components with Known Vulnerabilities
OWASP Top 10 2017: A9-Using Components with Known Vulnerabilities

Description

Dangerous Functions\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=40
Status	New

The dangerous function, memcpy, was found in use at line 60 in xLua/lj_dispatch.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	80	80
Object	memcpy	memcpy

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_init(GG_State *GG)

```
....
80.    memcpy(GG->got, dispatch_got, LJ_GOT__MAX*sizeof(ASMFunction *));
```

Dangerous Functions\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=41
Status	New

The dangerous function, memcpy, was found in use at line 106 in xLua/lj_dispatch.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	153	153

Object	memcpy	memcpy

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
153.        memcpy(&disp[0], &disp[GG_LEN_DDISP],
GG_LEN_SDISP*sizeof(ASMFunction));
```

Dangerous Functions\Path 3:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=42
Status New

The dangerous function, memcpy, was found in use at line 122 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	135	135
Object	memcpy	memcpy

Code Snippet
File Name xLua/lstrlib.c
Method static int str_rep (lua_State *L) {

```
....
135.        memcpy(p, s, l * sizeof(char)); p += l;
```

Dangerous Functions\Path 4:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=43
Status New

The dangerous function, memcpy, was found in use at line 122 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	137	137
Object	memcpy	memcpy

Code Snippet	
File Name	xLua/lstrlib.c
Method	static int str_rep (lua_State *L) {

```
....
137.          memcpy(p, sep, lsep * sizeof(char));
```

Dangerous Functions\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=44
Status	New

The dangerous function, memcpy, was found in use at line 122 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	141	141
Object	memcpy	memcpy

Code Snippet	
File Name	xLua/lstrlib.c
Method	static int str_rep (lua_State *L) {

```
....
141.      memcpy(p, s, l * sizeof(char));  /* last copy (not followed by
separator) */
```

Dangerous Functions\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=45
Status	New

The dangerous function, memcpy, was found in use at line 981 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	996	996
Object	memcpy	memcpy

Code Snippet

File Name	xLua/lstrlib.c
Method	static const char *scanformat (lua_State *L, const char *strfrmt, char *form) {

```
....
996.    memcpy(form, strfrmt, ((p - strfrmt) + 1) * sizeof(char));
```

Dangerous Functions\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=46
Status	New

The dangerous function, memcpy, was found in use at line 460 in xLua/lvm.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lvm.c	xLua/lvm.c
Line	464	464
Object	memcpy	memcpy

Code Snippet	
File Name	xLua/lvm.c
Method	static void copy2buff (StkId top, int n, char *buff) {

```
....
464.      memcpy(buff + tl, svalue(top - n), l * sizeof(char));
```

Dangerous Functions\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=47
Status	New

The dangerous function, strcat, was found in use at line 182 in xLua/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	196	196
Object	strcat	strcat

Code Snippet	
File Name	xLua/lobject.c
Method	void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
196.          strcat(out, "...");
```

Dangerous Functions\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=48
Status	New

The dangerous function, strcat, was found in use at line 182 in xLua/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	198	198
Object	strcat	strcat

Code Snippet
File Name xLua/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
198.        strcat(out, source);
```

Dangerous Functions\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=49
Status	New

The dangerous function, strcat, was found in use at line 182 in xLua/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	207	207
Object	strcat	strcat

Code Snippet
File Name xLua/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
207.          strcat(out, "...");
```

Dangerous Functions\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=50
Status	New

The dangerous function, strcat, was found in use at line 182 in xLua/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	210	210
Object	strcat	strcat

Code Snippet
File Name xLua/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
210.          strcat(out, source);
```

Dangerous Functions\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=51
Status	New

The dangerous function, strcat, was found in use at line 182 in xLua/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	211	211
Object	strcat	strcat

Code Snippet
File Name xLua/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
211.        strcat(out, "\"]");
```

Dangerous Functions\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=52
Status	New

The dangerous function, strcpy, was found in use at line 182 in xLua/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	193	193
Object	strcpy	strcpy

Code Snippet
File Name xLua/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
193.        strcpy(out, "");
```

Dangerous Functions\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=53
Status	New

The dangerous function, strcpy, was found in use at line 182 in xLua/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	204	204
Object	strcpy	strcpy

Code Snippet
File Name xLua/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
204.        strcpy(out, "[string \"");
```

Dangerous Functions\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=54
Status	New

The dangerous function, strcpy, was found in use at line 1006 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	1010	1010
Object	strcpy	strcpy

Code Snippet
File Name xLua/lstrlib.c
Method static void addlenmod (char *form, const char *lenmod) {

```
....
1010.    strcpy(form + l - 1, lenmod);
```

Dangerous Functions\Path 16:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=55
Status	New

The dangerous function, strlen, was found in use at line 182 in xLua/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	192	192
Object	strlen	strlen

Code Snippet
File Name xLua/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
192.        l = strlen(source);
```

Dangerous Functions\Path 17:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=56
Status	New

The dangerous function, strlen, was found in use at line 580 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	585	585
Object	strlen	strlen

Code Snippet
File Name xLua/lstrlib.c
Method static int nospecials (const char *p, size_t l) {

```
....
585.      upto += strlen(p + upto) + 1;  /* may have more after \0 */
```

Dangerous Functions\Path 18:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=57
Status	New

The dangerous function, strlen, was found in use at line 1006 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	1007	1007
Object	strlen	strlen

Code Snippet
File Name xLua/lstrlib.c
Method static void addlenmod (char *form, const char *lenmod) {

```
....
1007.    size_t l = strlen(form);
```

Dangerous Functions\Path 19:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=58
Status	New

The dangerous function, strlen, was found in use at line 1006 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	1008	1008
Object	strlen	strlen

Code Snippet
File Name xLua/lstrlib.c
Method static void addlenmod (char *form, const char *lenmod) {

```
....
1008.    size_t lm = strlen(lenmod);
```

Dangerous Functions\Path 20:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=59
Status	New

The dangerous function, strlen, was found in use at line 1016 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	1070	1070
Object	strlen	strlen

Code Snippet
File Name xLua/lstrlib.c
Method static int str_format (lua_State *L) {

```
....
1070.               luaL_argcheck(L, l == strlen(s), arg, "string
contains zeros");
```

Dangerous Functions\Path 21:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=60
Status	New

The dangerous function, strlen, was found in use at line 1330 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	1400	1400
Object	strlen	strlen

Code Snippet
File Name xLua/lstrlib.c
Method static int str_pack (lua_State *L) {

```
....
1400.           luaL_argcheck(L, strlen(s) == len, arg, "string contains
zeros");
```

Dangerous Functions\Path 22:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=61
Status	New

The dangerous function, strlen, was found in use at line 1476 in xLua/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	1524	1524
Object	strlen	strlen

Code Snippet
File Name xLua/lstrlib.c
Method static int str_unpack (lua_State *L) {

```
....
1524.          size_t len = (int)strlen(data + pos);
```

Dangerous Functions\Path 23:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=62
Status	New

The dangerous function, strlen, was found in use at line 248 in xLua/lvm.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lvm.c	xLua/lvm.c
Line	258	258
Object	strlen	strlen

Code Snippet
File Name xLua/lvm.c
Method static int l_strcmp (const TString *ls, const TString *rs) {

```
....
258.        size_t len = strlen(l);  /* index of first '\0' in both
strings */
```

Dangerous Functions\Path 24:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=63
Status	New

The dangerous function, strncat, was found in use at line 182 in xLua/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	206	206
Object	strncat	strncat

Code Snippet
File Name xLua/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
206.          strncat(out, source, len);
```

Dangerous Functions\Path 25:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=64
Status	New

The dangerous function, strncpy, was found in use at line 182 in xLua/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	184	184
Object	strncpy	strncpy

Code Snippet
File Name xLua/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
184.      strncpy(out, source+1, bufflen);  /* remove first char */
```

Buffer Overflow boundcpy WrongSizeParam

Query Path:
CPP\Cx\CPP Buffer Overflow\Buffer Overflow boundcpy WrongSizeParam Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
OWASP Top 10 2017: A1-Injection

Description

Buffer Overflow boundcpy WrongSizeParam\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=3
Status	New

The size of the buffer used by lj_dispatch_init in LJ_GOT__MAX, at line 60 of xLua/lj_dispatch.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that lj_dispatch_init passes to LJ_GOT__MAX, at line 60 of xLua/lj_dispatch.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c

Line	80	80
Object	LJ_GOT__MAX	LJ_GOT__MAX

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_init(GG_State *GG)

```
....
80.    memcpy(GG->got, dispatch_got, LJ_GOT__MAX*sizeof(ASMFunction *));
```

Buffer Overflow boundcpy WrongSizeParam\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=4
Status	New

The size of the buffer used by lj_dispatch_init in ASMFunction, at line 60 of xLua/lj_dispatch.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that lj_dispatch_init passes to ASMFunction, at line 60 of xLua/lj_dispatch.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	80	80
Object	ASMFunction	ASMFunction

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_init(GG_State *GG)

```
....
80.    memcpy(GG->got, dispatch_got, LJ_GOT__MAX*sizeof(ASMFunction *));
```

Buffer Overflow boundcpy WrongSizeParam\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=5
Status	New

The size of the buffer used by lj_dispatch_update in GG_LEN_SDISP, at line 106 of xLua/lj_dispatch.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that lj_dispatch_update passes to GG_LEN_SDISP, at line 106 of xLua/lj_dispatch.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c

Line	153	153
Object	GG_LEN_SDISP	GG_LEN_SDISP

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
153.        memcpy(&disp[0], &disp[GG_LEN_DDISP],
GG_LEN_SDISP*sizeof(ASMFunction));
```

Buffer Overflow boundcpy WrongSizeParam\Path 4:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=6
Status New

The size of the buffer used by lj_dispatch_update in ASMFunction, at line 106 of xLua/lj_dispatch.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that lj_dispatch_update passes to ASMFunction, at line 106 of xLua/lj_dispatch.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	153	153
Object	ASMFunction	ASMFunction

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
153.        memcpy(&disp[0], &disp[GG_LEN_DDISP],
GG_LEN_SDISP*sizeof(ASMFunction));
```

Buffer Overflow boundcpy WrongSizeParam\Path 5:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=7
Status New

The size of the buffer used by str_rep in l, at line 122 of xLua/lstrlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that str_rep passes to l, at line 122 of xLua/lstrlib.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c

Line	135	135
Object	l	l

Code Snippet
File Name xLua/lstrlib.c
Method static int str_rep (lua_State *L) {

```
....
135.        memcpy(p, s, l * sizeof(char)); p += l;
```

Buffer Overflow boundcpy WrongSizeParam\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=8
Status	New

The size of the buffer used by str_rep in char, at line 122 of xLua/lstrlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that str_rep passes to char, at line 122 of xLua/lstrlib.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	135	135
Object	char	char

Code Snippet
File Name xLua/lstrlib.c
Method static int str_rep (lua_State *L) {

```
....
135.        memcpy(p, s, l * sizeof(char)); p += l;
```

Buffer Overflow boundcpy WrongSizeParam\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=9
Status	New

The size of the buffer used by str_rep in lsep, at line 122 of xLua/lstrlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that str_rep passes to lsep, at line 122 of xLua/lstrlib.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	137	137

Object	lsep	lsep

Code Snippet
File Name xLua/lstrlib.c
Method static int str_rep (lua_State *L) {

```
....
137.          memcpy(p, sep, lsep * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=10
Status	New

The size of the buffer used by str_rep in char, at line 122 of xLua/lstrlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that str_rep passes to char, at line 122 of xLua/lstrlib.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	137	137
Object	char	char

Code Snippet
File Name xLua/lstrlib.c
Method static int str_rep (lua_State *L) {

```
....
137.          memcpy(p, sep, lsep * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=11
Status	New

The size of the buffer used by str_rep in l, at line 122 of xLua/lstrlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that str_rep passes to l, at line 122 of xLua/lstrlib.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	141	141
Object	l	l

Code Snippet	
File Name	xLua/lstrlib.c
Method	static int str_rep (lua_State *L) {

```
....
141.      memcpy(p, s, l * sizeof(char));  /* last copy (not followed by
separator) */
```

Buffer Overflow boundcpy WrongSizeParam\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=12
Status	New

The size of the buffer used by str_rep in char, at line 122 of xLua/lstrlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that str_rep passes to char, at line 122 of xLua/lstrlib.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	141	141
Object	char	char

Code Snippet	
File Name	xLua/lstrlib.c
Method	static int str_rep (lua_State *L) {

```
....
141.      memcpy(p, s, l * sizeof(char));  /* last copy (not followed by
separator) */
```

Buffer Overflow boundcpy WrongSizeParam\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=13
Status	New

The size of the buffer used by *scanformat in char, at line 981 of xLua/lstrlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *scanformat passes to char, at line 981 of xLua/lstrlib.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	996	996
Object	char	char

Code Snippet

File Name	xLua/lstrlib.c
Method	static const char *scanformat (lua_State *L, const char *strfrmt, char *form) {

```
....
996.    memcpy(form, strfrmt, ((p - strfrmt) + 1) * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=14
Status	New

The size of the buffer used by copy2buff in l, at line 460 of xLua/lvm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that copy2buff passes to l, at line 460 of xLua/lvm.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lvm.c	xLua/lvm.c
Line	464	464
Object	l	l

Code Snippet	
File Name	xLua/lvm.c
Method	static void copy2buff (StkId top, int n, char *buff) {

```
....
464.      memcpy(buff + tl, svalue(top - n), l * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=15
Status	New

The size of the buffer used by copy2buff in char, at line 460 of xLua/lvm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that copy2buff passes to char, at line 460 of xLua/lvm.c, to overwrite the target buffer.

	Source	Destination
File	xLua/lvm.c	xLua/lvm.c
Line	464	464
Object	char	char

Code Snippet	
File Name	xLua/lvm.c
Method	static void copy2buff (StkId top, int n, char *buff) {

```
....
464.      memcpy(buff + tl, svalue(top - n), l * sizeof(char));
```

Unchecked Array Index

Query Path:
CPP\Cx\CPP Low Visibility\Unchecked Array Index Version:1

Categories

NIST SP 800-53: SI-10 Information Input Validation (P1)

Description

Unchecked Array Index\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=19
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	69	69
Object	BC_FORL	BC_FORL

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_init(GG_State *GG)

```
....
69.    disp[BC_FORL] = disp[BC_IFORL];
```

Unchecked Array Index\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=20
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	70	70
Object	BC_ITERL	BC_ITERL

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_init(GG_State *GG)

```
....
70.     disp[BC_ITERL] = disp[BC_IITERL];
```

Unchecked Array Index\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=21
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	72	72
Object	BC_ITERN	BC_ITERN

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_init(GG_State *GG)

```
....
72.     disp[BC_ITERN] = &lj_vm_IITERN;
```

Unchecked Array Index\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=22
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	73	73
Object	BC_LOOP	BC_LOOP

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_init(GG_State *GG)

```
....
73.     disp[BC_LOOP] = disp[BC_ILOOP];
```

Unchecked Array Index\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=23
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	156	156
Object	BC_RETM	BC_RETM

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
156.          disp[BC_RETM] = lj_vm_rethook;
```

Unchecked Array Index\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=24
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	157	157
Object	BC_RET	BC_RET

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
157.          disp[BC_RET] = lj_vm_rethook;
```

Unchecked Array Index\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=25
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	158	158

Object	BC_RET0	BC_RET0

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
158.          disp[BC_RET0] = lj_vm_rethook;
```

Unchecked Array Index\Path 8:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=26
Status New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	159	159
Object	BC_RET1	BC_RET1

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
159.          disp[BC_RET1] = lj_vm_rethook;
```

Unchecked Array Index\Path 9:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=27
Status New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	171	171
Object	BC_FORL	BC_FORL

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
171.        disp[BC_FORL] = f_forl;
```

Unchecked Array Index\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=28
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	172	172
Object	BC_ITERL	BC_ITERL

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
172.        disp[BC_ITERL] = f_iterl;
```

Unchecked Array Index\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=29
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	173	173
Object	BC_ITERN	BC_ITERN

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
173.        disp[BC_ITERN] = f_itern;
```

Unchecked Array Index\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://WIN-

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=30
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	174	174
Object	BC_LOOP	BC_LOOP

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
174.        disp[BC_LOOP] = f_loop;
```

Unchecked Array Index\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=31
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	177	177
Object	BC_RETM	BC_RETM

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
177.        disp[BC_RETM] = lj_vm_rethook;
```

Unchecked Array Index\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=32
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	178	178

Object	BC_RET	BC_RET

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
178.        disp[BC_RET] = lj_vm_rethook;
```

Unchecked Array Index\Path 15:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=33
Status New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	179	179
Object	BC_RET0	BC_RET0

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
179.        disp[BC_RET0] = lj_vm_rethook;
```

Unchecked Array Index\Path 16:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=34
Status New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	180	180
Object	BC_RET1	BC_RET1

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
180.        disp[BC_RET1] = lj_vm_rethook;
```

Unchecked Array Index\Path 17:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=35
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	182	182
Object	BC_RETM	BC_RETM

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
182.        disp[BC_RETM] = disp[GG_LEN_DDISP+BC_RETM];
```

Unchecked Array Index\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=36
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	183	183
Object	BC_RET	BC_RET

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
183.        disp[BC_RET] = disp[GG_LEN_DDISP+BC_RET];
```

Unchecked Array Index\Path 19:

Severity	Low
Result State	To Verify
Online Results	http://WIN-

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=37
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	184	184
Object	BC_RET0	BC_RET0

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
184.        disp[BC_RET0] = disp[GG_LEN_DDISP+BC_RET0];
```

Unchecked Array Index\Path 20:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=38
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	185	185
Object	BC_RET1	BC_RET1

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_update(global_State *g)

```
....
185.        disp[BC_RET1] = disp[GG_LEN_DDISP+BC_RET1];
```

Unchecked Array Index\Path 21:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=39
Status	New

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	836	836

Object	n	n

Code Snippet
File Name xLua/lstrlib.c
Method static lua_Number adddigit (char *buff, int n, lua_Number x) {

```
....
836.    buff[n] = (d < 10 ? d + '0' : d - 10 + 'a');  /* add to buffer
*/
```

Improper Resource Access Authorization

Query Path:
CPP\Cx\CPP Low Visibility\Improper Resource Access Authorization Version:1

Categories

FISMA 2014: Identification And Authentication
NIST SP 800-53: AC-3 Access Enforcement (P1)
OWASP Top 10 2017: A2-Broken Authentication

Description

Improper Resource Access Authorization\Path 1:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=65
Status New

	Source	Destination
File	xLua/test.cpp	xLua/test.cpp
Line	65	65
Object	fprintf	fprintf

Code Snippet
File Name xLua/test.cpp
Method main(int argc, char** argv)

```
....
65.    fprintf(stdout, "======================\nRun unit
testing...\n");
```

Improper Resource Access Authorization\Path 2:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=66
Status New

	Source	Destination

File	xLua/test.cpp	xLua/test.cpp
Line	70	70
Object	fprintf	fprintf

Code Snippet
File Name xLua/test.cpp
Method main(int argc, char** argv)

```
....
70.    fprintf(stdout, TestErrMsgMgr::noError() ? "succ\n\n" :
"fail\n\n");
```

Improper Resource Access Authorization\Path 3:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=67
Status New

	Source	Destination
File	xLua/test.cpp	xLua/test.cpp
Line	13	13
Object	fprintf	fprintf

Code Snippet
File Name xLua/test.cpp
Method smoke_test()

```
....
13.    fprintf(stdout, "running smoke tests...\n");
```

Improper Resource Access Authorization\Path 4:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=68
Status New

	Source	Destination
File	xLua/test.cpp	xLua/test.cpp
Line	34	34
Object	fprintf	fprintf

Code Snippet
File Name xLua/test.cpp

Method	verify_log2()

```
....
34.    fprintf(stdout, "verify log2...\n");
```

Improper Resource Access Authorization\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=69
Status	New

	Source	Destination
File	xLua/test.cpp	xLua/test.cpp
Line	55	55
Object	fprintf	fprintf

Code Snippet
File Name xLua/test.cpp
Method verify_log2()

```
....
55.        fprintf(stderr, "log2(%u) expect %u, get %u\n", v,
log2_expect, log2_get);
```

Use of Sizeof On a Pointer Type

Query Path:
CPP\Cx\CPP Low Visibility\Use of Sizeof On a Pointer Type Version:1

Description

Use of Sizeof On a Pointer Type\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=1
Status	New

	Source	Destination
File	xLua/lj_dispatch.c	xLua/lj_dispatch.c
Line	80	80
Object	sizeof	sizeof

Code Snippet
File Name xLua/lj_dispatch.c
Method void lj_dispatch_init(GG_State *GG)

```
....
80.    memcpy(GG->got, dispatch_got, LJ_GOT__MAX*sizeof(ASMFunction *));
```

Use of Sizeof On a Pointer Type\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=2
Status	New

	Source	Destination
File	xLua/lobject.c	xLua/lobject.c
Line	144	144
Object	sizeof	sizeof

Code Snippet
File Name xLua/lobject.c
Method const char *luaO_pushvfstring (lua_State *L, const char *fmt, va_list argp) {

```
....
144.          char buff[4*sizeof(void *) + 8]; /* should be enough space
for a `%p' */
```

Arithmenic Operation On Boolean

Query Path:
CPP\Cx\CPP Low Visibility\Arithmenic Operation On Boolean Version:1

Categories

FISMA 2014: Audit And Accountability
NIST SP 800-53: SC-5 Denial of Service Protection (P1)

Description

Arithmenic Operation On Boolean\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=17
Status	New

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	128	128
Object	BinaryExpr	BinaryExpr

Code Snippet
File Name xLua/lstrlib.c

Method	static int str_rep (lua_State *L) {

```
....
128.    else if (l + lsep < l || l + lsep > MAXSIZE / n)  /* may
overflow? */
```

Arithmenic Operation On Boolean\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=18
Status	New

	Source	Destination
File	xLua/lstrlib.c	xLua/lstrlib.c
Line	1426	1426
Object	BinaryExpr	BinaryExpr

Code Snippet
File Name xLua/lstrlib.c
Method static int str_packsize (lua_State *L) {

```
....
1426.      luaL_argcheck(L, totalsize <= MAXSIZE - size, 1,
```

NULL Pointer Dereference

Query Path:
CPP\Cx\CPP Low Visibility\NULL Pointer Dereference Version:1

Categories

NIST SP 800-53: SC-5 Denial of Service Protection (P1)
OWASP Top 10 2017: A1-Injection

Description

NULL Pointer Dereference\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1020016&projectid=20012&pathid=16
Status	New

The variable declared in 0 at xLua/lgc.c in line 125 is not initialized when it is used by g at xLua/lgc.c in line 654.

	Source	Destination
File	xLua/lgc.c	xLua/lgc.c
Line	137	657

Object	0	g

Code Snippet
File Name xLua/lgc.c
Method static GCObject **getgclist (GCObject *o) {

```
....
137.      default: lua_assert(0); return 0;
```

File Name xLua/lgc.c
Method static lu_mem propagatemark (global_State *g) {

```
....
657.    g->gray = *getgclist(o);  /* remove from 'gray' list */
```

Buffer Overflow boundcpy WrongSizeParam

Risk

What might happen

Buffer overflow attacks, in their various forms, could allow an attacker to control certain areas of memory. Typically, this is used to overwrite data on the stack necessary for the program to function properly, such as code and memory addresses, though other forms of this attack exist. Exploiting this vulnerability can generally lead to system crashes, infinite loops, or even execution of arbitrary code.

Cause

How does it happen

Buffer Overflows can manifest in numerous different variations. In it's most basic form, the attack controls a buffer, which is then copied to a smaller buffer without size verification. Because the attacker's source buffer is larger than the program's target buffer, the attacker's data overwrites whatever is next on the stack, allowing the attacker to control program structures.
Alternatively, the vulnerability could be the result of improper bounds checking; exposing internal memory addresses outside of their valid scope; allowing the attacker to control the size of the target buffer; or various other forms.

General Recommendations

How to avoid it

- Always perform proper bounds checking before copying buffers or strings.
- Prefer to use safer functions and structures, e.g. safe string classes over `char*`, `strncpy` over `strcpy`, and so on.
- Consistently apply tests for the size of buffers.
- Do not return variable addresses outside the scope of their variables.

Source Code Examples

CPP

Overflowing Buffers

```
const int BUFFER_SIZE = 10;
char buffer[BUFFER_SIZE];

void copyStringToBuffer(char* inputString)
{
    strcpy(buffer, inputString);
}
```

Checked Buffers

```
const int BUFFER_SIZE = 10;
const int MAX_INPUT_SIZE = 256;
char buffer[BUFFER_SIZE];

void copyStringToBuffer(char* inputString)
{
    if (strnlen(inputString, MAX_INPUT_SIZE) < sizeof(buffer))
    {
        strncpy(buffer, inputString, sizeof(buffer));
    }
}
```

Dangerous Functions

Risk

What might happen

Use of dangerous functions may expose varying risks associated with each particular function, with potential impact of improper usage of these functions varying significantly. The presence of such functions indicates a flaw in code maintenance policies and adherence to secure coding practices, in a way that has allowed introducing known dangerous code into the application.

Cause

How does it happen

A dangerous function has been identified within the code. Functions are often deemed dangerous to use for numerous reasons, as there are different sets of vulnerabilities associated with usage of such functions. For example, some string copy and concatenation functions are vulnerable to Buffer Overflow, Memory Disclosure, Denial of Service and more. Use of these functions is not recommended.

General Recommendations

How to avoid it

- Deploy a secure and recommended alternative to any functions that were identified as dangerous.
    - If no secure alternative is found, conduct further researching and testing to identify whether current usage successfully sanitizes and verifies values, and thus successfully avoids the use-cases for whom the function is indeed dangerous
- Conduct a periodical review of methods that are in use, to ensure that all external libraries and built-in functions are up-to-date and whose use has not been excluded from best secure coding practices.

Source Code Examples

CPP

Buffer Overflow in gets()

```
int main()
{
	char buf[10];

	printf("Please enter your name: ");
	gets(buf); // veryveryverylongname
	if (buf == ACCEPTED_NAME)
	{
		// Do something
	}
	return 0;
}
```

Safe reading from user

```
int main()
{
	char buf[10];

	printf("Please enter your name: ");
	fgets(buf, sizeof(buf), stdin); //setting the amount of bytes to read
	if (buf == ACCEPTED_NAME)
	{
		//Do something
	}
	return 0;
}
```

Unsafe function for string copy

```
int main(int argc, char* argv[])
{
	char buf[10];
	strcpy(buf, argv[1]); // overflow occurs when len(argv[1]) > 10 bytes

	return 0;
}
```

Safe string copy

```
int main(int argc, char* argv[])
{
	char buf[10];
	strncpy(buf, argv[1], sizeof(buf));
	buf[9]= '\0'; //strncpy doesn't NULL terminates

	return 0;
}
```

Unsafe format string

```
int main(int argc, char* argv[])
{
	printf(argv[1]); // If argv[1] contains a format token, such as %s,%x or %d, will cause
an access violation
	return 0;
}
```

Safe format string

```
int main(int argc, char* argv[])
{
      printf("%s", argv[1]); // Second parameter is not a formattable string
      return 0;
}
```

Use of sizeof() on a Pointer Type

Weakness ID: 467 *(Weakness Variant)* **Status:** Draft

Description

Description Summary

The code calls sizeof() on a malloced pointer type, which always returns the wordsize/8. This can produce an unexpected result if the programmer intended to determine how much memory has been allocated.

Time of Introduction

- Implementation

Applicable Platforms

Languages

C

C++

Common Consequences

Scope	Effect
Integrity	This error can often cause one to allocate a buffer that is much smaller than what is needed, leading to resultant weaknesses such as buffer overflows.

Likelihood of Exploit

High

Demonstrative Examples

Example 1

Care should be taken to ensure sizeof returns the size of the data structure itself, and not the size of the pointer to the data structure.

In this example, sizeof(foo) returns the size of the pointer.

(Bad Code)

Example Languages: **C and C++**

```
double *foo;
...
foo = (double *)malloc(sizeof(foo));
```

In this example, sizeof(*foo) returns the size of the data structure and not the size of the pointer.

(Good Code)

Example Languages: **C and C++**

```
double *foo;
...
foo = (double *)malloc(sizeof(*foo));
```

Example 2

This example defines a fixed username and password. The AuthenticateUser() function is intended to accept a username and a password from an untrusted user, and check to ensure that it matches the username and password. If the username and password match, AuthenticateUser() is intended to indicate that authentication succeeded.

(Bad Code)

```
/* Ignore CWE-259 (hard-coded password) and CWE-309 (use of password system for authentication) for this example. */

char *username = "admin";
char *pass = "password";

int AuthenticateUser(char *inUser, char *inPass) {
```

```
printf("Sizeof username = %d\n", sizeof(username));
printf("Sizeof pass = %d\n", sizeof(pass));

if (strncmp(username, inUser, sizeof(username))) {
printf("Auth failure of username using sizeof\n");
return(AUTH_FAIL);
}
/* Because of CWE-467, the sizeof returns 4 on many platforms and architectures. */
if (! strncmp(pass, inPass, sizeof(pass))) {
printf("Auth success of password using sizeof\n");
return(AUTH_SUCCESS);
}
else {
printf("Auth fail of password using sizeof\n");
return(AUTH_FAIL);
}
}

int main (int argc, char **argv)
{
int authResult;

if (argc < 3) {
ExitError("Usage: Provide a username and password");
}
authResult = AuthenticateUser(argv[1], argv[2]);
if (authResult != AUTH_SUCCESS) {
ExitError("Authentication failed");
}
else {
DoAuthenticatedTask(argv[1]);
}
}
```

In AuthenticateUser(), because sizeof() is applied to a parameter with an array type, the sizeof() call might return 4 on many modern architectures. As a result, the strncmp() call only checks the first four characters of the input password, resulting in a partial comparison (CWE-187), leading to improper authentication (CWE-287).

Because of the partial comparison, any of these passwords would still cause authentication to succeed for the "admin" user:

(Attack)

```
pass5
passABCDEFGH
passWORD
```

Because only 4 characters are checked, this significantly reduces the search space for an attacker, making brute force attacks more feasible.

The same problem also applies to the username, so values such as "adminXYZ" and "administrator" will succeed for the username.

Potential Mitigations

Phase: Implementation

Use expressions such as "sizeof(*pointer)" instead of "sizeof(pointer)", unless you intend to run sizeof() on a pointer type to gain some platform independence or if you are allocating a variable on the stack.

Other Notes

The use of sizeof() on a pointer can sometimes generate useful information. An obvious case is to find out the wordsize on a platform. More often than not, the appearance of sizeof(pointer) indicates a bug.

Weakness Ordinalities

Ordinality	Description
Primary	*(where the weakness exists independent of other weaknesses)*

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	465	Pointer Issues	**Development Concepts (primary)699**
ChildOf	Weakness Class	682	Incorrect Calculation	**Research Concepts (primary)1000**
ChildOf	Category	737	CERT C Secure Coding Section 03 - Expressions (EXP)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
ChildOf	Category	740	CERT C Secure Coding Section 06 - Arrays (ARR)	Weaknesses Addressed by the CERT C Secure Coding Standard734
CanPrecede	Weakness Base	131	Incorrect Calculation of Buffer Size	Research Concepts1000

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
CLASP			Use of sizeof() on a pointer type
CERT C Secure Coding	ARR01-C		Do not apply the sizeof operator to a pointer when taking the size of an array
CERT C Secure Coding	EXP01-C		Do not take the size of a pointer to determine the size of the pointed-to type

White Box Definitions

A weakness where code path has:

1. end statement that passes an identity of a dynamically allocated memory resource to a sizeof operator
2. start statement that allocates the dynamically allocated memory resource

References

Robert Seacord. "EXP01-A. Do not take the sizeof a pointer to determine the size of a type". <https://www.securecoding.cert.org/confluence/display/seccode/EXP01-A.+Do+not+take+the+sizeof+a+pointer+to+determine+the+size+of+a+type>.

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	CLASP		Externally Mined
Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-07-01	Eric Dalci	Cigital	External
	updated Time of Introduction		
2008-08-01		KDM Analytics	External
	added/updated white box definitions		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Relationships, Other Notes, Taxonomy Mappings, Weakness Ordinalities		
2008-11-24	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		
2009-12-28	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Relationships		

BACK TO TOP

NULL Pointer Dereference

Risk

What might happen

A null pointer dereference is likely to cause a run-time exception, a crash, or other unexpected behavior.

Cause

How does it happen

Variables which are declared without being assigned will implicitly retain a null value until they are assigned. The null value can also be explicitly set to a variable, to ensure clear out its contents. Since null is not really a value, it may not have object variables and methods, and any attempt to access contents of a null object, instead of verifying it is set beforehand, will result in a null pointer dereference exception.

General Recommendations

How to avoid it

- For any variable that is created, ensure all logic flows between declaration and use assign a non-null value to the variable first.
- Enforce null checks on any received variable or object before it is dereferenced, to ensure it does not contain a null assigned to it elsewhere.
- Consider the need to assign null values in order to overwrite initialized variables. Consider reassigning or releasing these variables instead.

Source Code Examples

CPP

Explicit NULL Dereference

```
char * input = NULL;
printf("%s", input);
```

Implicit NULL Dereference

```
char * input;
printf("%s", input);
```

Java

Explicit Null Dereference

```
Object o = null;
out.println(o.getClass());
```

Indicator of Poor Code Quality

Weakness ID: 398 *(Weakness Class)* **Status:** Draft

Description

Description Summary

The code has features that do not directly introduce a weakness or vulnerability, but indicate that the product has not been carefully developed or maintained.

Extended Description

Programs are more likely to be secure when good development practices are followed. If a program is complex, difficult to maintain, not portable, or shows evidence of neglect, then there is a higher likelihood that weaknesses are buried in the code.

Time of Introduction

- Architecture and Design
- Implementation

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	18	Source Code	**Development Concepts (primary)699**
ChildOf	Weakness Class	710	Coding Standards Violation	**Research Concepts (primary)1000**
ParentOf	Weakness Variant	107	Struts: Unused Validation Form	**Research Concepts (primary)1000**
ParentOf	Weakness Variant	110	Struts: Validator Without Form Field	**Research Concepts (primary)1000**
ParentOf	Category	399	Resource Management Errors	**Development Concepts (primary)699**
ParentOf	Weakness Base	401	Failure to Release Memory Before Removing Last Reference ('Memory Leak')	**Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Base	404	Improper Resource Shutdown or Release	Development Concepts699 **Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Variant	415	Double Free	**Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Base	416	Use After Free	**Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Variant	457	Use of Uninitialized Variable	**Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Base	474	Use of Function with Inconsistent Implementations	**Development Concepts (primary)699 Seven Pernicious Kingdoms (primary)700 Research Concepts (primary)1000**
ParentOf	Weakness Base	475	Undefined Behavior for Input to API	**Development Concepts (primary)699 Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Base	476	NULL Pointer Dereference	**Development Concepts**

				(primary)699 Seven Pernicious Kingdoms (primary)700 Research Concepts (primary)1000
ParentOf	Weakness Base	477	Use of Obsolete Functions	**Development Concepts (primary)699 Seven Pernicious Kingdoms (primary)700 Research Concepts (primary)1000**
ParentOf	Weakness Variant	478	Missing Default Case in Switch Statement	**Development Concepts (primary)699**
ParentOf	Weakness Variant	479	Unsafe Function Call from a Signal Handler	**Development Concepts (primary)699**
ParentOf	Weakness Variant	483	Incorrect Block Delimitation	**Development Concepts (primary)699**
ParentOf	Weakness Base	484	Omitted Break Statement in Switch	**Development Concepts (primary)699** Research Concepts1000
ParentOf	Weakness Variant	546	Suspicious Comment	**Development Concepts (primary)699 Research Concepts (primary)1000**
ParentOf	Weakness Variant	547	Use of Hard-coded, Security-relevant Constants	**Development Concepts (primary)699 Research Concepts (primary)1000**
ParentOf	Weakness Variant	561	Dead Code	**Development Concepts (primary)699 Research Concepts (primary)1000**
ParentOf	Weakness Base	562	Return of Stack Variable Address	**Development Concepts (primary)699** Research Concepts1000
ParentOf	Weakness Variant	563	Unused Variable	**Development Concepts (primary)699 Research Concepts (primary)1000**
ParentOf	Category	569	Expression Issues	**Development Concepts (primary)699**
ParentOf	Weakness Variant	585	Empty Synchronized Block	**Development Concepts (primary)699 Research Concepts (primary)1000**
ParentOf	Weakness Variant	586	Explicit Call to Finalize()	**Development Concepts (primary)699**
ParentOf	Weakness Variant	617	Reachable Assertion	**Development Concepts (primary)699**
ParentOf	Weakness Base	676	Use of Potentially Dangerous Function	**Development Concepts (primary)699 Research Concepts (primary)1000**
MemberOf	View	700	Seven Pernicious Kingdoms	**Seven Pernicious Kingdoms (primary)700**

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name

7 Pernicious Kingdoms			Code Quality

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	7 Pernicious Kingdoms		Externally Mined
Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-07-01	Eric Dalci	Cigital	External
	updated Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Description, Relationships, Taxonomy Mappings		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Relationships		
Previous Entry Names			
Change Date	**Previous Entry Name**		
2008-04-11	Code Quality		

BACK TO TOP

Improper Validation of Array Index

Weakness ID: 129 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The product uses untrusted input when calculating or using an array index, but the product does not validate or incorrectly validates the index to ensure the index references a valid position within the array.

Alternate Terms

out-of-bounds array index

index-out-of-range

array index underflow

Time of Introduction

- Implementation

Applicable Platforms

Languages

C: *(Often)*

C++: *(Often)*

Language-independent

Common Consequences

Scope	Effect
Integrity Availability	Unchecked array indexing will very likely result in the corruption of relevant memory and perhaps instructions, leading to a crash, if the values are outside of the valid memory area.
Integrity	If the memory corrupted is data, rather than instructions, the system will continue to function with improper values.
Confidentiality Integrity	Unchecked array indexing can also trigger out-of-bounds read or write operations, or operations on the wrong objects; i.e., "buffer overflows" are not always the result. This may result in the exposure or modification of sensitive data.
Integrity	If the memory accessible by the attacker can be effectively controlled, it may be possible to execute arbitrary code, as with a standard buffer overflow and possibly without the use of large inputs if a precise index can be controlled.
Integrity Availability Confidentiality	A single fault could allow either an overflow (CWE-788) or underflow (CWE-786) of the array index. What happens next will depend on the type of operation being performed out of bounds, but can expose sensitive information, cause a system crash, or possibly lead to arbitrary code execution.

Likelihood of Exploit

High

Detection Methods

Automated Static Analysis

This weakness can often be detected using automated static analysis tools. Many modern tools use data flow analysis or constraint-based techniques to minimize the number of false positives.

Automated static analysis generally does not account for environmental considerations when reporting out-of-bounds memory operations. This can make it difficult for users to determine which warnings should be investigated first. For example, an analysis tool might report array index errors that originate from command line arguments in a program that is not expected to run with setuid or other special privileges.

Effectiveness: High

This is not a perfect solution, since 100% accuracy and coverage are not feasible.

Automated Dynamic Analysis

This weakness can be detected using dynamic tools and techniques that interact with the software using large test suites with many diverse inputs, such as fuzz testing (fuzzing), robustness testing, and fault injection. The software's operation may slow down, but it should not become unstable, crash, or generate incorrect results.

Black Box

Black box methods might not get the needed code coverage within limited time constraints, and a dynamic test might not produce any noticeable side effects even if it is successful.

Demonstrative Examples

Example 1

The following C/C++ example retrieves the sizes of messages for a pop3 mail server. The message sizes are retrieved from a socket that returns in a buffer the message number and the message size, the message number (num) and size (size) are extracted from the buffer and the message size is placed into an array using the message number for the array index.

(Bad Code)

Example Language: **C**

```
/* capture the sizes of all messages */
int getsizes(int sock, int count, int *sizes) {
...
char buf[BUFFER_SIZE];
int ok;
int num, size;

// read values from socket and added to sizes array
while ((ok = gen_recv(sock, buf, sizeof(buf))) == 0)
{

// continue read from socket until buf only contains '.'
if (DOTLINE(buf))
break;
else if (sscanf(buf, "%d %d", &num, &size) == 2)
sizes[num - 1] = size;
}
...
}
```

In this example the message number retrieved from the buffer could be a value that is outside the allowable range of indices for the array and could possibly be a negative number. Without proper validation of the value to be used for the array index an array overflow could occur and could potentially lead to unauthorized access to memory addresses and system crashes. The value of the array index should be validated to ensure that it is within the allowable range of indices for the array as in the following code.

(Good Code)

Example Language: **C**

```
/* capture the sizes of all messages */
int getsizes(int sock, int count, int *sizes) {
...
char buf[BUFFER_SIZE];
int ok;
int num, size;

// read values from socket and added to sizes array
while ((ok = gen_recv(sock, buf, sizeof(buf))) == 0)
{

// continue read from socket until buf only contains '.'
if (DOTLINE(buf))
```

```
break;
else if (sscanf(buf, "%d %d", &num, &size) == 2) {
if (num > 0 && num <= (unsigned)count)
sizes[num - 1] = size;
else
/* warn about possible attempt to induce buffer overflow */
report(stderr, "Warning: ignoring bogus data for message sizes returned by server.\n");
}
}
...
}
```

Example 2

In the code snippet below, an unchecked integer value is used to reference an object in an array.

(Bad Code)

Example Language: **Java**

```
public String getValue(int index) {
return array[index];
}
```

If index is outside of the range of the array, this may result in an ArrayIndexOutOfBounds Exception being raised.

Example 3

In the following Java example the method displayProductSummary is called from a Web service servlet to retrieve product summary information for display to the user. The servlet obtains the integer value of the product number from the user and passes it to the displayProductSummary method. The displayProductSummary method passes the integer value of the product number to the getProductSummary method which obtains the product summary from the array object containing the project summaries using the integer value of the product number as the array index.

(Bad Code)

Example Language: **Java**

```
// Method called from servlet to obtain product information
public String displayProductSummary(int index) {

String productSummary = new String("");

try {
String productSummary = getProductSummary(index);

} catch (Exception ex) {...}

return productSummary;
}

public String getProductSummary(int index) {
return products[index];
}
```

In this example the integer value used as the array index that is provided by the user may be outside the allowable range of indices for the array which may provide unexpected results or may comes the application to fail. The integer value used for the array index should be validated to ensure that it is within the allowable range of indices for the array as in the following code.

(Good Code)

Example Language: **Java**

```
// Method called from servlet to obtain product information
public String displayProductSummary(int index) {

String productSummary = new String("");
```

```
try {
String productSummary = getProductSummary(index);

} catch (Exception ex) {...}

return productSummary;
}

public String getProductSummary(int index) {
String productSummary = "";

if ((index >= 0) && (index < MAX_PRODUCTS)) {
productSummary = products[index];
}
else {
System.err.println("index is out of bounds");
throw new IndexOutOfBoundsException();
}

return productSummary;
}
```

An alternative in Java would be to use one of the collection objects such as ArrayList that will automatically generate an exception if an attempt is made to access an array index that is out of bounds.

(Good Code)

Example Language: **Java**

```
ArrayList productArray = new ArrayList(MAX_PRODUCTS);
...
try {
productSummary = (String) productArray.get(index);
} catch (IndexOutOfBoundsException ex) {...}
```

Observed Examples

Reference	Description
CVE-2005-0369	large ID in packet used as array index
CVE-2001-1009	negative array index as argument to POP LIST command
CVE-2003-0721	Integer signedness error leads to negative array index
CVE-2004-1189	product does not properly track a count and a maximum number, which can lead to resultant array index overflow.
CVE-2007-5756	chain: device driver for packet-capturing software allows access to an unintended IOCTL with resultant array index error.

Potential Mitigations

Phase: Architecture and Design

Strategies: Input Validation; Libraries or Frameworks

Use an input validation framework such as Struts or the OWASP ESAPI Validation API. If you use Struts, be mindful of weaknesses covered by the CWE-101 category.

Phase: Architecture and Design

For any security checks that are performed on the client side, ensure that these checks are duplicated on the server side, in order to avoid CWE-602. Attackers can bypass the client-side checks by modifying values after the checks have been performed, or by changing the client to remove the client-side checks entirely. Then, these modified values would be submitted to the server.

Even though client-side checks provide minimal benefits with respect to server-side security, they are still useful. First, they can support intrusion detection. If the server receives input that should have been rejected by the client, then it may be an indication of an attack. Second, client-side error-checking can provide helpful feedback to the user about the expectations for valid input. Third, there may be a reduction in server-side processing time for accidental input errors, although this is typically a small savings.

Phase: Requirements

Strategy: Language Selection

Use a language with features that can automatically mitigate or eliminate out-of-bounds indexing errors.

For example, Ada allows the programmer to constrain the values of a variable and languages such as Java and Ruby will allow the programmer to handle exceptions when an out-of-bounds index is accessed.

Phase: Implementation

Strategy: Input Validation

Assume all input is malicious. Use an "accept known good" input validation strategy (i.e., use a whitelist). Reject any input that does not strictly conform to specifications, or transform it into something that does. Use a blacklist to reject any unexpected inputs and detect potential attacks.

When accessing a user-controlled array index, use a stringent range of values that are within the target array. Make sure that you do not allow negative values to be used. That is, verify the minimum as well as the maximum of the range of acceptable values.

Phase: Implementation

Be especially careful to validate your input when you invoke code that crosses language boundaries, such as from an interpreted language to native code. This could create an unexpected interaction between the language boundaries. Ensure that you are not violating any of the expectations of the language with which you are interfacing. For example, even though Java may not be susceptible to buffer overflows, providing a large argument in a call to native code might trigger an overflow.

Weakness Ordinalities

Ordinality	Description
Resultant	The most common condition situation leading to unchecked array indexing is the use of loop index variables as buffer indexes. If the end condition for the loop is subject to a flaw, the index can grow or shrink unbounded, therefore causing a buffer overflow or underflow. Another common situation leading to this condition is the use of a function's return value, or the resulting value of a calculation directly as an index in to a buffer.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	20	Improper Input Validation	**Development Concepts (primary)699 Research Concepts (primary)1000**
ChildOf	Category	189	Numeric Errors	Development Concepts699
ChildOf	Category	633	Weaknesses that Affect Memory	**Resource-specific Weaknesses (primary)631**
ChildOf	Category	738	CERT C Secure Coding Section 04 - Integers (INT)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
ChildOf	Category	740	CERT C Secure Coding Section 06 - Arrays (ARR)	Weaknesses Addressed by the CERT C Secure Coding Standard734
ChildOf	Category	802	2010 Top 25 - Risky Resource Management	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**
CanPrecede	Weakness Class	119	Failure to Constrain Operations within the Bounds of a Memory Buffer	Research Concepts1000
CanPrecede	Weakness Variant	789	Uncontrolled Memory Allocation	Research Concepts1000
PeerOf	Weakness Base	124	Buffer Underwrite ('Buffer Underflow')	Research Concepts1000

Theoretical Notes

An improperly validated array index might lead directly to the always-incorrect behavior of "access of array using out-of-bounds index."

Affected Resources

- Memory

f Causal Nature

Explicit

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
CLASP			Unchecked array indexing
PLOVER			INDEX - Array index overflow
CERT C Secure Coding	ARR00-C		Understand how arrays work
CERT C Secure Coding	ARR30-C		Guarantee that array indices are within the valid range
CERT C Secure Coding	ARR38-C		Do not add or subtract an integer to a pointer if the resulting value does not refer to a valid array element
CERT C Secure Coding	INT32-C		Ensure that operations on signed integers do not result in overflow

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
100	Overflow Buffers	

References

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 5, "Array Indexing Errors" Page 144. 2nd Edition. Microsoft. 2002.

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	CLASP		Externally Mined
Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-07-01	Sean Eidemiller	Cigital	External
	added/updated demonstrative examples		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Alternate Terms, Applicable Platforms, Common Consequences, Relationships, Other Notes, Taxonomy Mappings, Weakness Ordinalities		
2008-11-24	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2009-01-12	CWE Content Team	MITRE	Internal
	updated Common Consequences		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Description, Name, Relationships		
2009-12-28	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Observed Examples, Other Notes, Potential Mitigations, Theoretical Notes, Weakness Ordinalities		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Demonstrative Examples, Detection Factors, Likelihood of Exploit, Potential Mitigations, References, Related Attack Patterns, Relationships		
2010-04-05	CWE Content Team	MITRE	Internal
	updated Related Attack Patterns		
Previous Entry Names			
Change Date	**Previous Entry Name**		
2009-10-29	Unchecked Array Indexing		

BACK TO TOP

Improper Access Control (Authorization)

Weakness ID: 285 *(Weakness Class)* **Status:** Draft

Description

Description Summary

The software does not perform or incorrectly performs access control checks across all potential execution paths.

Extended Description

When access control checks are not applied consistently - or not at all - users are able to access data or perform actions that they should not be allowed to perform. This can lead to a wide range of problems, including information leaks, denial of service, and arbitrary code execution.

Alternate Terms

AuthZ:	"AuthZ" is typically used as an abbreviation of "authorization" within the web application security community. It is also distinct from "AuthC," which is an abbreviation of "authentication." The use of "Auth" as an abbreviation is discouraged, since it could be used for either authentication or authorization.

Time of Introduction

- Architecture and Design
- Implementation
- Operation

Applicable Platforms

Languages

Language-independent

Technology Classes

Web-Server: *(Often)*

Database-Server: *(Often)*

Modes of Introduction

A developer may introduce authorization weaknesses because of a lack of understanding about the underlying technologies. For example, a developer may assume that attackers cannot modify certain inputs such as headers or cookies.

Authorization weaknesses may arise when a single-user application is ported to a multi-user environment.

Common Consequences

Scope	Effect
Confidentiality	An attacker could read sensitive data, either by reading the data directly from a data store that is not properly restricted, or by accessing insufficiently-protected, privileged functionality to read the data.
Integrity	An attacker could modify sensitive data, either by writing the data directly to a data store that is not properly restricted, or by accessing insufficiently-protected, privileged functionality to write the data.
Integrity	An attacker could gain privileges by modifying or reading critical data directly, or by accessing insufficiently-protected, privileged functionality.

Likelihood of Exploit

High

Detection Methods

Automated Static Analysis

Automated static analysis is useful for detecting commonly-used idioms for authorization. A tool may be able to analyze related configuration files, such as .htaccess in Apache web servers, or detect the usage of commonly-used authorization libraries.

Generally, automated static analysis tools have difficulty detecting custom authorization schemes. In addition, the software's design may include some functionality that is accessible to any user and does not require an authorization check; an automated technique that detects the absence of authorization may report false positives.

Effectiveness: Limited

Automated Dynamic Analysis

Automated dynamic analysis may find many or all possible interfaces that do not require authorization, but manual analysis is required to determine if the lack of authorization violates business logic

Manual Analysis

This weakness can be detected using tools and techniques that require manual (human) analysis, such as penetration testing, threat modeling, and interactive tools that allow the tester to record and modify an active session.

Specifically, manual static analysis is useful for evaluating the correctness of custom authorization mechanisms.

Effectiveness: Moderate

These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules. However, manual efforts might not achieve desired code coverage within limited time constraints.

Demonstrative Examples

Example 1

The following program could be part of a bulletin board system that allows users to send private messages to each other. This program intends to authenticate the user before deciding whether a private message should be displayed. Assume that LookupMessageObject() ensures that the $id argument is numeric, constructs a filename based on that id, and reads the message details from that file. Also assume that the program stores all private messages for all users in the same directory.

(Bad Code)

Example Language: **Perl**

```
sub DisplayPrivateMessage {
my($id) = @_;
my $Message = LookupMessageObject($id);
print "From: " . encodeHTML($Message->{from}) . "<br>\n";
print "Subject: " . encodeHTML($Message->{subject}) . "\n";
print "<hr>\n";
print "Body: " . encodeHTML($Message->{body}) . "\n";
}

my $q = new CGI;
# For purposes of this example, assume that CWE-309 and
# CWE-523 do not apply.
if (! AuthenticateUser($q->param('username'), $q->param('password'))) {
ExitError("invalid username or password");
}

my $id = $q->param('id');
DisplayPrivateMessage($id);
```

While the program properly exits if authentication fails, it does not ensure that the message is addressed to the user. As a result, an authenticated attacker could provide any arbitrary identifier and read private messages that were intended for other users.

One way to avoid this problem would be to ensure that the "to" field in the message object matches the username of the authenticated user.

Observed Examples

Reference	Description
CVE-2009-3168	Web application does not restrict access to admin scripts, allowing authenticated users to reset administrative passwords.

CVE-2009-2960	Web application does not restrict access to admin scripts, allowing authenticated users to modify passwords of other users.
CVE-2009-3597	Web application stores database file under the web root with insufficient access control (CWE-219), allowing direct request.
CVE-2009-2282	Terminal server does not check authorization for guest access.
CVE-2009-3230	Database server does not use appropriate privileges for certain sensitive operations.
CVE-2009-2213	Gateway uses default "Allow" configuration for its authorization settings.
CVE-2009-0034	Chain: product does not properly interpret a configuration option for a system group, allowing users to gain privileges.
CVE-2008-6123	Chain: SNMP product does not properly parse a configuration option for which hosts are allowed to connect, allowing unauthorized IP addresses to connect.
CVE-2008-5027	System monitoring software allows users to bypass authorization by creating custom forms.
CVE-2008-7109	Chain: reliance on client-side security (CWE-602) allows attackers to bypass authorization using a custom client.
CVE-2008-3424	Chain: product does not properly handle wildcards in an authorization policy list, allowing unintended access.
CVE-2009-3781	Content management system does not check access permissions for private files, allowing others to view those files.
CVE-2008-4577	ACL-based protection mechanism treats negative access rights as if they are positive, allowing bypass of intended restrictions.
CVE-2008-6548	Product does not check the ACL of a page accessed using an "include" directive, allowing attackers to read unauthorized files.
CVE-2007-2925	Default ACL list for a DNS server does not set certain ACLs, allowing unauthorized DNS queries.
CVE-2006-6679	Product relies on the X-Forwarded-For HTTP header for authorization, allowing unintended access by spoofing the header.
CVE-2005-3623	OS kernel does not check for a certain privilege before setting ACLs for files.
CVE-2005-2801	Chain: file-system code performs an incorrect comparison (CWE-697), preventing defauls ACLs from being properly applied.
CVE-2001-1155	Chain: product does not properly check the result of a reverse DNS lookup because of operator precedence (CWE-783), allowing bypass of DNS-based access restrictions.

Potential Mitigations

Phase: Architecture and Design

Divide your application into anonymous, normal, privileged, and administrative areas. Reduce the attack surface by carefully mapping roles with data and functionality. Use role-based access control (RBAC) to enforce the roles at the appropriate boundaries.

Note that this approach may not protect against horizontal authorization, i.e., it will not protect a user from attacking others with the same role.

Phase: Architecture and Design

Ensure that you perform access control checks related to your business logic. These checks may be different than the access control checks that you apply to more generic resources such as files, connections, processes, memory, and database records. For example, a database may restrict access for medical records to a specific database user, but each record might only be intended to be accessible to the patient and the patient's doctor.

Phase: Architecture and Design

Strategy: Libraries or Frameworks

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness

easier to avoid.

For example, consider using authorization frameworks such as the JAAS Authorization Framework and the OWASP ESAPI Access Control feature.

Phase: Architecture and Design

For web applications, make sure that the access control mechanism is enforced correctly at the server side on every page. Users should not be able to access any unauthorized functionality or information by simply requesting direct access to that page.

One way to do this is to ensure that all pages containing sensitive information are not cached, and that all such pages restrict access to requests that are accompanied by an active and authenticated session token associated with a user who has the required permissions to access that page.

Phases: System Configuration; Installation

Use the access control capabilities of your operating system and server environment and define your access control lists accordingly. Use a "default deny" policy when defining these ACLs.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	254	Security Features	**Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Class	284	Access Control (Authorization) Issues	**Development Concepts (primary)699 Research Concepts (primary)1000**
ChildOf	Category	721	OWASP Top Ten 2007 Category A10 - Failure to Restrict URL Access	**Weaknesses in OWASP Top Ten (2007) (primary)629**
ChildOf	Category	723	OWASP Top Ten 2004 Category A2 - Broken Access Control	**Weaknesses in OWASP Top Ten (2004) (primary)711**
ChildOf	Category	753	2009 Top 25 - Porous Defenses	**Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)750**
ChildOf	Category	803	2010 Top 25 - Porous Defenses	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**
ParentOf	Weakness Variant	219	Sensitive Data Under Web Root	**Research Concepts (primary)1000**
ParentOf	Weakness Base	551	Incorrect Behavior Order: Authorization Before Parsing and Canonicalization	**Development Concepts (primary)699** Research Concepts1000
ParentOf	Weakness Class	638	Failure to Use Complete Mediation	Research Concepts1000
ParentOf	Weakness Base	804	Guessable CAPTCHA	**Development Concepts (primary)699 Research Concepts (primary)1000**

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Missing Access Control
OWASP Top Ten 2007	A10	CWE More Specific	Failure to Restrict URL Access
OWASP Top Ten 2004	A2	CWE More Specific	Broken Access Control

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
1	Accessing Functionality Not Properly Constrained by ACLs	
13	Subverting Environment Variable Values	

17	Accessing, Modifying or Executing Executable Files
87	Forceful Browsing
39	Manipulating Opaque Client-based Data Tokens
45	Buffer Overflow via Symbolic Links
51	Poison Web Service Registry
59	Session Credential Falsification through Prediction
60	Reusing Session IDs (aka Session Replay)
77	Manipulating User-Controlled Variables
76	Manipulating Input to File System Calls
104	Cross Zone Scripting

References

NIST. "Role Based Access Control and Role Based Security". <http://csrc.nist.gov/groups/SNS/rbac/>.

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 4, "Authorization" Page 114; Chapter 6, "Determining Appropriate Access Control" Page 171. 2nd Edition. Microsoft. 2002.

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	7 Pernicious Kingdoms		Externally Mined
Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-07-01	Eric Dalci	Cigital	External
	updated Time of Introduction		
2008-08-15		Veracode	External
	Suggested OWASP Top Ten 2004 mapping		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Relationships, Other Notes, Taxonomy Mappings		
2009-01-12	CWE Content Team	MITRE	Internal
	updated Common Consequences, Description, Likelihood of Exploit, Name, Other Notes, Potential Mitigations, References, Relationships		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Potential Mitigations		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Description, Related Attack Patterns		
2009-07-27	CWE Content Team	MITRE	Internal
	updated Relationships		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Type		
2009-12-28	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Demonstrative Examples, Detection Factors, Modes of Introduction, Observed Examples, Relationships		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Alternate Terms, Detection Factors, Potential Mitigations, References, Relationships		
2010-04-05	CWE Content Team	MITRE	Internal
	updated Potential Mitigations		
Previous Entry Names			
Change Date	**Previous Entry Name**		
2009-01-12	Missing or Inconsistent Access Control		

BACK TO TOP

Scanned Languages

Language	Hash Number	Change Date
CPP	4541647240435660	6/19/2024
Common	0105849645654507	6/19/2024