# civetweb Scan Report

| | |
|---|---|
| Project Name | civetweb |
| Scan Start | Thursday, June 20, 2024 2:06:22 PM |
| Preset | Checkmarx Default |
| Scan Time | 00h:12m:40s |
| Lines Of Code Scanned | 31829 |
| Files Scanned | 62 |
| Report Creation Time | Thursday, June 20, 2024 3:14:08 PM |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3 |
| Team | CxServer |
| Checkmarx Version | 8.7.0 |
| Scan Type | Full |
| Source Origin | LocalPath |
| Density | 4/1000 (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 |

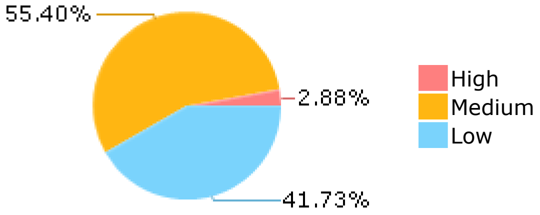
**Results Limit**

Results limit per query was set to 50

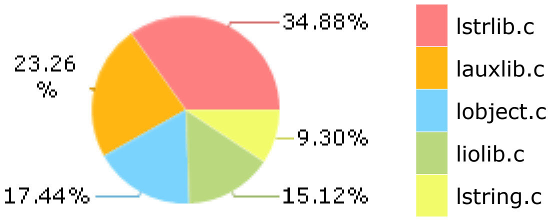
**Selected Queries**

Selected queries are listed in Result Summary

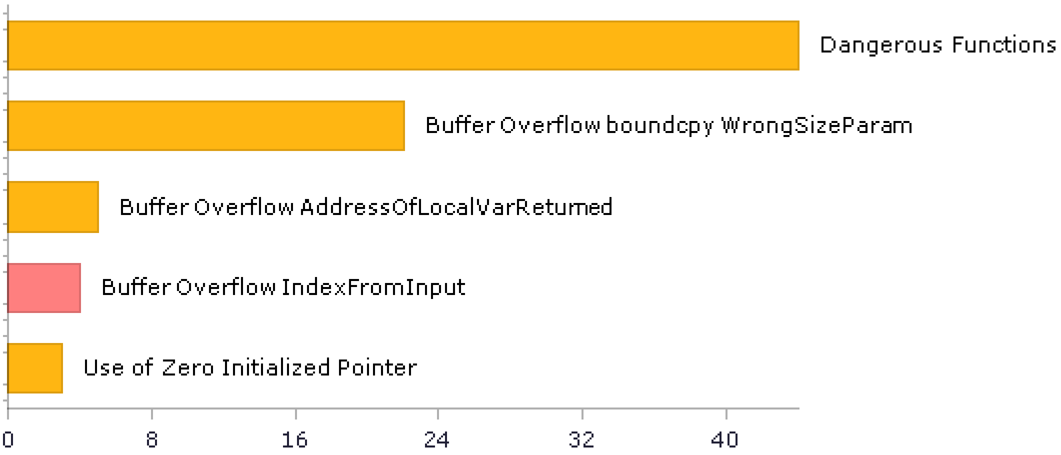
## Result Summary



## 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 | 36 | 32 |
| A2-Broken Authentication | App. Specific | EASY | COMMON | AVERAGE | SEVERE | App. Specific | 21 | 21 |
| A3-Sensitive Data Exposure | App. Specific | AVERAGE | WIDESPREAD | AVERAGE | SEVERE | App. Specific | 5 | 5 |
| 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 | 44 | 44 |
| 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 | 44 | 44 |
| 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 | 2 | 2 |
| PCI DSS (3.2) - 6.5.2 - Buffer overflows | 29 | 29 |
| 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. | 1 | 1 |
| 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. | 0 | 0 |
| 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. | 2 | 2 |
| 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. | 20 | 20 |
| 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. | 3 | 3 |
| 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. | 2 | 2 |

***** 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) | 22 | 22 |
| 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) | 3 | 3 |
| SC-4 Information in Shared Resources (P1) | 2 | 2 |
| SC-5 Denial of Service Protection (P1)***** | 11 | 10 |
| SC-8 Transmission Confidentiality and Integrity (P1) | 1 | 1 |
| SI-10 Information Input Validation (P1)***** | 6 | 6 |
| SI-11 Error Handling (P2)***** | 9 | 9 |
| SI-15 Information Output Filtering (P0)***** | 0 | 0 |
| SI-16 Memory Protection (P1) | 2 | 2 |

***** 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:<br>-Failing to identify the user at all when that should be required<br>-Failure to maintain the user's identity when it is required<br>-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.).<br>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 |

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

# Scan Summary - Custom

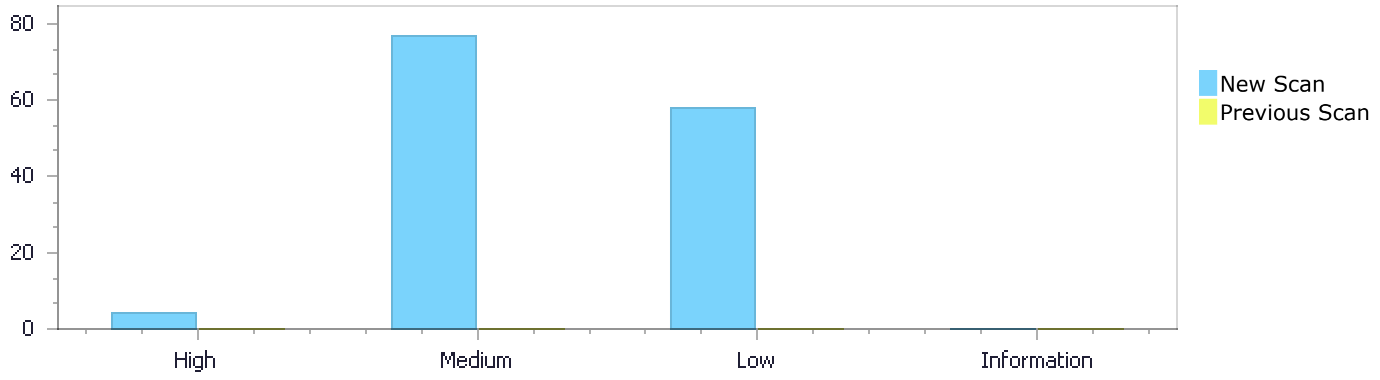
| Category | Issues Found | Best Fix Locations |
|---|---|---|
| Must audit | 0 | 0 |
| Check | 0 | 0 |
| Optional | 0 | 0 |

## Results Distribution By Status

First scan of the project

| | High | Medium | Low | Information | Total |
|---|---|---|---|---|---|
| New Issues | 4 | 77 | 58 | 0 | 139 |
| Recurrent Issues | 0 | 0 | 0 | 0 | 0 |
| Total | 4 | 77 | 58 | 0 | 139 |

| | | | | | |
|---|---|---|---|---|---|
| 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 | 4 | 77 | 58 | 0 | 139 |
| Urgent | 0 | 0 | 0 | 0 | 0 |
| Proposed Not Exploitable | 0 | 0 | 0 | 0 | 0 |
| Total | 4 | 77 | 58 | 0 | 139 |

## Result Summary

| Vulnerability Type | Occurrences | Severity |
|---|---|---|
| Buffer Overflow IndexFromInput | 4 | High |
| Dangerous Functions | 44 | Medium |
| Buffer Overflow boundcpy WrongSizeParam | 22 | Medium |
| Buffer Overflow AddressOfLocalVarReturned | 5 | Medium |
| Use of Zero Initialized Pointer | 3 | Medium |

| | | |
|---|---|---|
| Integer Overflow | 2 | Medium |
| Wrong Size t Allocation | 1 | Medium |
| Improper Resource Access Authorization | 20 | Low |
| Unchecked Return Value | 9 | Low |
| TOCTOU | 5 | Low |
| Use of Sizeof On a Pointer Type | 5 | Low |
| Unchecked Array Index | 4 | Low |
| NULL Pointer Dereference | 3 | Low |
| Use of Insufficiently Random Values | 3 | Low |
| Insecure Temporary File | 2 | Low |
| Potential Off by One Error in Loops | 2 | Low |
| Sizeof Pointer Argument | 2 | Low |
| Client Insufficient ClickJacking Protection | 1 | Low |
| Exposure of System Data to Unauthorized Control Sphere | 1 | Low |
| Incorrect Permission Assignment For Critical Resources | 1 | Low |

# 10 Most Vulnerable Files

## High and Medium Vulnerabilities

| File Name | Issues Found |
|---|---|
| lua-5.2.4/src/lstrlib.c | 23 |
| lua-5.2.4/src/lauxlib.c | 17 |
| lua-5.2.4/src/lobject.c | 14 |
| lua-5.2.4/src/lstring.c | 6 |
| lua-5.2.4/src/ltable.c | 4 |
| lua-5.2.4/src/lundump.c | 4 |
| lua-5.2.4/src/lua.c | 3 |
| lua-5.2.4/src/lparser.c | 3 |
| lua-5.2.4/src/lzio.c | 2 |
| lua-5.2.4/src/liolib.c | 2 |

# Scan Results Details

## Buffer Overflow IndexFromInput

Query Path:
CPP\Cx\CPP Buffer Overflow\Buffer Overflow IndexFromInput Version:1

### Categories

OWASP Top 10 2017: A1-Injection

*Description*

**Buffer Overflow IndexFromInput\Path 1:**

| | |
|---|---|
| Severity | High |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=136 |
| Status | New |

The size of the buffer used by luaL_loadfilex in PostfixExpr, at line 630 of lua-5.2.4/src/lauxlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that skipBOM passes to getc, at line 596 of lua-5.2.4/src/lauxlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 601 | 653 |
| Object | getc | PostfixExpr |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method static int skipBOM (LoadF *lf) {

```
....
601.        c = getc(lf->f);
```

File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API int luaL_loadfilex (lua_State *L, const char *filename,

```
....
653.        lf.buff[lf.n++] = c;  /* 'c' is the first character of the
stream */
```

**Buffer Overflow IndexFromInput\Path 2:**

| | |
|---|---|
| Severity | High |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=137 |
| Status | New |

The size of the buffer used by luaL_loadfilex in PostfixExpr, at line 630 of lua-5.2.4/src/lauxlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaL_loadfilex passes to stdin, at line 630 of lua-5.2.4/src/lauxlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 638 | 653 |
| Object | stdin | PostfixExpr |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API int luaL_loadfilex (lua_State *L, const char *filename,

```
....
638.        lf.f = stdin;
....
653.        lf.buff[lf.n++] = c;  /* 'c' is the first character of the
stream */
```

**Buffer Overflow IndexFromInput\Path 3:**

| | |
|---|---|
| Severity | High |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=138 |
| Status | New |

The size of the buffer used by luaL_loadfilex in PostfixExpr, at line 630 of lua-5.2.4/src/lauxlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that skipBOM passes to getc, at line 596 of lua-5.2.4/src/lauxlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 601 | 646 |
| Object | getc | PostfixExpr |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method static int skipBOM (LoadF *lf) {

```
....
601.        c = getc(lf->f);
```

File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API int luaL_loadfilex (lua_State *L, const char *filename,

```
....
646.        lf.buff[lf.n++] = '\n';  /* add line to correct line numbers
*/
```

**Buffer Overflow IndexFromInput\Path 4:**

| | |
|---|---|
| Severity | High |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=139](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=139) |
| Status | New |

The size of the buffer used by luaL_loadfilex in PostfixExpr, at line 630 of lua-5.2.4/src/lauxlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaL_loadfilex passes to stdin, at line 630 of lua-5.2.4/src/lauxlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 638 | 646 |
| Object | stdin | PostfixExpr |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API int luaL_loadfilex (lua_State *L, const char *filename,

```
....
638.        lf.f = stdin;
....
646.        lf.buff[lf.n++] = '\n';   /* add line to correct line numbers
*/
```

## 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=1000007&projectid=3&pathid=62](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=62) |
| Status | New |

The dangerous function, memcpy, was found in use at line 437 in lua-5.2.4/src/lauxlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 449 | 449 |
| Object | memcpy | memcpy |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API char *luaL_prepbuffsize (luaL_Buffer *B, size_t sz) {

```
....
449.        memcpy(newbuff, B->b, B->n * sizeof(char));
```

**Dangerous Functions\Path 2:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=63 |
| Status | New |

The dangerous function, memcpy, was found in use at line 459 in lua-5.2.4/src/lauxlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 461 | 461 |
| Object | memcpy | memcpy |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API void luaL_addlstring (luaL_Buffer *B, const char *s, size_t l) {

```
....
461.      memcpy(b, s, l * sizeof(char));
```

**Dangerous Functions\Path 3:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=64 |
| Status | New |

The dangerous function, memcpy, was found in use at line 252 in lua-5.2.4/src/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 256 | 256 |
| Object | memcpy | memcpy |

Code Snippet
File Name lua-5.2.4/src/lobject.c

| | |
|---|---|
| Method | void luaO_chunkid (char *out, const char *source, size_t bufflen) { |

```
....
256.        memcpy(out, source + 1, l * sizeof(char));
```

**Dangerous Functions\Path 4:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=65](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=65) |
| Status | New |

The dangerous function, memcpy, was found in use at line 252 in lua-5.2.4/src/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 264 | 264 |
| Object | memcpy | memcpy |

| | |
|---|---|
| Code Snippet | |
| File Name | lua-5.2.4/src/lobject.c |
| Method | void luaO_chunkid (char *out, const char *source, size_t bufflen) { |

```
....
264.        memcpy(out, source + 1, l * sizeof(char));
```

**Dangerous Functions\Path 5:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=66](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=66) |
| Status | New |

The dangerous function, memcpy, was found in use at line 252 in lua-5.2.4/src/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 268 | 268 |
| Object | memcpy | memcpy |

| | |
|---|---|
| Code Snippet | |
| File Name | lua-5.2.4/src/lobject.c |
| Method | void luaO_chunkid (char *out, const char *source, size_t bufflen) { |

```
....
268.        memcpy(out, source + 1 + l - bufflen, bufflen *
sizeof(char));
```

**Dangerous Functions\Path 6:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=67 |
| Status | New |

The dangerous function, memcpy, was found in use at line 252 in lua-5.2.4/src/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 284 | 284 |
| Object | memcpy | memcpy |

| | |
|---|---|
| Code Snippet | |
| File Name | lua-5.2.4/src/lobject.c |
| Method | void luaO_chunkid (char *out, const char *source, size_t bufflen) { |

```
....
284.      memcpy(out, POS, (LL(POS) + 1) * sizeof(char));
```

**Dangerous Functions\Path 7:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=68 |
| Status | New |

The dangerous function, memcpy, was found in use at line 98 in lua-5.2.4/src/lstring.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstring.c | lua-5.2.4/src/lstring.c |
| Line | 107 | 107 |
| Object | memcpy | memcpy |

| | |
|---|---|
| Code Snippet | |
| File Name | lua-5.2.4/src/lstring.c |
| Method | static TString *createstrobj (lua_State *L, const char *str, size_t l, |

```
....
107.    memcpy(ts+1, str, l*sizeof(char));
```

### Dangerous Functions\Path 8:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=69 |
| Status | New |

The dangerous function, memcpy, was found in use at line 108 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 121 | 121 |
| Object | memcpy | memcpy |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static int str_rep (lua_State *L) { |

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

### Dangerous Functions\Path 9:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=70 |
| Status | New |

The dangerous function, memcpy, was found in use at line 108 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 123 | 123 |
| Object | memcpy | memcpy |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static int str_rep (lua_State *L) { |

```
....
123.          memcpy(p, sep, lsep * sizeof(char)); p += lsep;
```

**Dangerous Functions\Path 10:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=71](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=71) |
| Status | New |

The dangerous function, memcpy, was found in use at line 108 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 126 | 126 |
| Object | memcpy | memcpy |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static int str_rep (lua_State *L) { |

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

**Dangerous Functions\Path 11:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=72](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=72) |
| Status | New |

The dangerous function, memcpy, was found in use at line 857 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 872 | 872 |
| Object | memcpy | memcpy |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static const char *scanformat (lua_State *L, const char *strfrmt, char *form) { |

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

**Dangerous Functions\Path 12:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=73 |
| Status | New |

The dangerous function, memcpy, was found in use at line 244 in lua-5.2.4/src/lundump.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lundump.c | lua-5.2.4/src/lundump.c |
| Line | 247 | 247 |
| Object | memcpy | memcpy |

| | |
|---|---|
| Code Snippet | |
| File Name | lua-5.2.4/src/lundump.c |
| Method | void luaU_header (lu_byte* h) |

```
....
247.   memcpy(h,LUA_SIGNATURE,sizeof(LUA_SIGNATURE)-sizeof(char));
```

**Dangerous Functions\Path 13:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=74 |
| Status | New |

The dangerous function, memcpy, was found in use at line 244 in lua-5.2.4/src/lundump.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lundump.c | lua-5.2.4/src/lundump.c |
| Line | 257 | 257 |
| Object | memcpy | memcpy |

| | |
|---|---|
| Code Snippet | |
| File Name | lua-5.2.4/src/lundump.c |
| Method | void luaU_header (lu_byte* h) |

```
....
257.    memcpy(h,LUAC_TAIL,sizeof(LUAC_TAIL)-sizeof(char));
```

**Dangerous Functions\Path 14:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=75 |
| Status | New |

The dangerous function, memcpy, was found in use at line 190 in lua-5.2.4/src/lundump.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lundump.c | lua-5.2.4/src/lundump.c |
| Line | 195 | 195 |
| Object | memcpy | memcpy |

Code Snippet
File Name lua-5.2.4/src/lundump.c
Method static void LoadHeader(LoadState* S)

```
....
195.    memcpy(s,h,sizeof(char));                  /* first char already
read */
```

**Dangerous Functions\Path 15:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=76 |
| Status | New |

The dangerous function, memcpy, was found in use at line 46 in lua-5.2.4/src/lzio.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lzio.c | lua-5.2.4/src/lzio.c |
| Line | 58 | 58 |
| Object | memcpy | memcpy |

Code Snippet
File Name lua-5.2.4/src/lzio.c
Method size_t luaZ_read (ZIO *z, void *b, size_t n) {

```
....
58.      memcpy(b, z->p, m);
```

**Dangerous Functions\Path 16:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=77 |
| Status | New |

The dangerous function, sprintf, was found in use at line 833 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 845 | 845 |
| Object | sprintf | sprintf |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static void addquoted (lua_State *L, luaL_Buffer *b, int arg) {

```
....
845.          sprintf(buff, "\\%d", (int)uchar(*s));
```

**Dangerous Functions\Path 17:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=78 |
| Status | New |

The dangerous function, sprintf, was found in use at line 833 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 847 | 847 |
| Object | sprintf | sprintf |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static void addquoted (lua_State *L, luaL_Buffer *b, int arg) {

```
....
847.          sprintf(buff, "\\%03d", (int)uchar(*s));
```

**Dangerous Functions\Path 18:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=79](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=79) |
| Status | New |

The dangerous function, sprintf, was found in use at line 892 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 914 | 914 |
| Object | sprintf | sprintf |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static int str_format (lua_State *L) { |

```
....
914.            nb = sprintf(buff, form, luaL_checkint(L, arg));
```

**Dangerous Functions\Path 19:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=80](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=80) |
| Status | New |

The dangerous function, sprintf, was found in use at line 892 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 924 | 924 |
| Object | sprintf | sprintf |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static int str_format (lua_State *L) { |

```
....
924.                nb = sprintf(buff, form, ni);
```

**Dangerous Functions\Path 20:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=81 |
| Status | New |

The dangerous function, sprintf, was found in use at line 892 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 934 | 934 |
| Object | sprintf | sprintf |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static int str_format (lua_State *L) {

```
....
934.                nb = sprintf(buff, form, ni);
```

**Dangerous Functions\Path 21:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=82 |
| Status | New |

The dangerous function, sprintf, was found in use at line 892 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 943 | 943 |
| Object | sprintf | sprintf |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static int str_format (lua_State *L) {

```
....
943.            nb = sprintf(buff, form,
(LUA_FLTFRM_T)luaL_checknumber(L, arg));
```

### Dangerous Functions\Path 22:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=83 |
| Status | New |

The dangerous function, sprintf, was found in use at line 892 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 960 | 960 |
| Object | sprintf | sprintf |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static int str_format (lua_State *L) {

```
....
960.              nb = sprintf(buff, form, s);
```

### Dangerous Functions\Path 23:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=84 |
| Status | New |

The dangerous function, strcpy, was found in use at line 105 in lua-5.2.4/src/loslib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/loslib.c | lua-5.2.4/src/loslib.c |
| Line | 108 | 108 |
| Object | strcpy | strcpy |

Code Snippet
File Name lua-5.2.4/src/loslib.c
Method static int os_tmpname (lua_State *L) {

```
....
108.     lua_tmpnam(buff, err);
```

**Dangerous Functions\Path 24:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=85 |
| Status | New |

The dangerous function, strcpy, was found in use at line 882 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 886 | 886 |
| Object | strcpy | strcpy |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static void addlenmod (char *form, const char *lenmod) {

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

**Dangerous Functions\Path 25:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=86 |
| Status | New |

The dangerous function, strlen, was found in use at line 364 in lua-5.2.4/src/lauxlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 368 | 368 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API const char *luaL_optlstring (lua_State *L, int narg,

```
....
368.          *len = (def ? strlen(def) : 0);
```

**Dangerous Functions\Path 26:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=87 |
| Status | New |

The dangerous function, strlen, was found in use at line 466 in lua-5.2.4/src/lauxlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 467 | 467 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API void luaL_addstring (luaL_Buffer *B, const char *s) {

```
....
467.     luaL_addlstring(B, s, strlen(s));
```

**Dangerous Functions\Path 27:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=88 |
| Status | New |

The dangerous function, strlen, was found in use at line 691 in lua-5.2.4/src/lauxlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 692 | 692 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API int luaL_loadstring (lua_State *L, const char *s) {

```
....
692.    return luaL_loadbuffer(L, s, strlen(s), s);
```

**Dangerous Functions\Path 28:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=89](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=89) |
| Status | New |

The dangerous function, strlen, was found in use at line 767 in lua-5.2.4/src/lauxlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 773 | 773 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method static const char *luaL_findtable (lua_State *L, int idx,

```
....
773.      if (e == NULL) e = fname + strlen(fname);
```

**Dangerous Functions\Path 29:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=90](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=90) |
| Status | New |

The dangerous function, strlen, was found in use at line 902 in lua-5.2.4/src/lauxlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 905 | 905 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API const char *luaL_gsub (lua_State *L, const char *s, const char *p,

```
....
905.    size_t l = strlen(p);
```

### Dangerous Functions\Path 30:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=91](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=91) |
| Status | New |

The dangerous function, strlen, was found in use at line 354 in lua-5.2.4/src/ldblib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ldblib.c | lua-5.2.4/src/ldblib.c |
| Line | 361 | 361 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/ldblib.c
Method static int db_debug (lua_State *L) {

```
....
361.      if (luaL_loadbuffer(L, buffer, strlen(buffer), "=(debug
command)")  ||
```

### Dangerous Functions\Path 31:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=92](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=92) |
| Status | New |

The dangerous function, strlen, was found in use at line 263 in lua-5.2.4/src/liolib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 268 | 268 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/liolib.c
Method static FILE *getiofile (lua_State *L, const char *findex) {

```
....
268.        luaL_error(L, "standard %s file is closed", findex +
strlen(IO_PREFIX));
```

### Dangerous Functions\Path 32:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=93 |
| Status | New |

The dangerous function, strlen, was found in use at line 371 in lua-5.2.4/src/liolib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 381 | 381 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/liolib.c
Method static int read_line (lua_State *L, FILE *f, int chop) {

```
....
381.        l = strlen(p);
```

### Dangerous Functions\Path 33:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=94 |
| Status | New |

The dangerous function, strlen, was found in use at line 338 in lua-5.2.4/src/loadlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/loadlib.c | lua-5.2.4/src/loadlib.c |
| Line | 343 | 343 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/loadlib.c
Method static const char *pushnexttemplate (lua_State *L, const char *path) {

```
....
343.     if (l == NULL) l = path + strlen(path);
```

**Dangerous Functions\Path 34:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=95 |
| Status | New |

The dangerous function, strlen, was found in use at line 252 in lua-5.2.4/src/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 253 | 253 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
253.     size_t l = strlen(source);
```

**Dangerous Functions\Path 35:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=96 |
| Status | New |

The dangerous function, strlen, was found in use at line 179 in lua-5.2.4/src/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 190 | 190 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lobject.c
Method const char *luaO_pushvfstring (lua_State *L, const char *fmt, va_list argp) {

```
....
190.          pushstr(L, s, strlen(s));
```

**Dangerous Functions\Path 36:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=97](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=97) |
| Status | New |

The dangerous function, strlen, was found in use at line 179 in lua-5.2.4/src/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 227 | 227 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lobject.c
Method const char *luaO_pushvfstring (lua_State *L, const char *fmt, va_list argp) {

```
....
227.     pushstr(L, fmt, strlen(fmt));
```

**Dangerous Functions\Path 37:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=98](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=98) |
| Status | New |

The dangerous function, strlen, was found in use at line 170 in lua-5.2.4/src/lstring.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstring.c | lua-5.2.4/src/lstring.c |
| Line | 171 | 171 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lstring.c
Method TString *luaS_new (lua_State *L, const char *str) {

```
....
171.     return luaS_newlstr(L, str, strlen(str));
```

**Dangerous Functions\Path 38:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=99 |
| Status | New |

The dangerous function, strlen, was found in use at line 566 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 571 | 571 |
| Object | strlen | strlen |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static int nospecials (const char *p, size_t l) { |

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

**Dangerous Functions\Path 39:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=100 |
| Status | New |

The dangerous function, strlen, was found in use at line 882 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 883 | 883 |
| Object | strlen | strlen |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static void addlenmod (char *form, const char *lenmod) { |

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

### Dangerous Functions\Path 40:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=101 |
| Status | New |

The dangerous function, strlen, was found in use at line 882 in lua-5.2.4/src/lstrlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 884 | 884 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static void addlenmod (char *form, const char *lenmod) {

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

### Dangerous Functions\Path 41:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=102 |
| Status | New |

The dangerous function, strlen, was found in use at line 186 in lua-5.2.4/src/lua.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lua.c | lua-5.2.4/src/lua.c |
| Line | 187 | 187 |
| Object | strlen | strlen |

Code Snippet
File Name lua-5.2.4/src/lua.c
Method static void print_version (void) {

```
....
187.    luai_writestring(LUA_COPYRIGHT, strlen(LUA_COPYRIGHT));
```

**Dangerous Functions\Path 42:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=103](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=103) |
| Status | New |

The dangerous function, strlen, was found in use at line 217 in lua-5.2.4/src/lua.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lua.c | lua-5.2.4/src/lua.c |
| Line | 218 | 218 |
| Object | strlen | strlen |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lua.c |
| Method | static int dostring (lua_State *L, const char *s, const char *name) { |

```
....
218.    int status = luaL_loadbuffer(L, s, strlen(s), name);
```

**Dangerous Functions\Path 43:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=104](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=104) |
| Status | New |

The dangerous function, strlen, was found in use at line 260 in lua-5.2.4/src/lua.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lua.c | lua-5.2.4/src/lua.c |
| Line | 269 | 269 |
| Object | strlen | strlen |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lua.c |
| Method | static int pushline (lua_State *L, int firstline) { |

```
....
269.    l = strlen(b);
```

**Dangerous Functions\Path 44:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=105](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=105) |
| Status | New |

The dangerous function, realloc, was found in use at line 919 in lua-5.2.4/src/lauxlib.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 926 | 926 |
| Object | realloc | realloc |

Code Snippet
File Name    lua-5.2.4/src/lauxlib.c
Method    static void *l_alloc (void *ud, void *ptr, size_t osize, size_t nsize) {

```
....
926.      return realloc(ptr, nsize);
```

## 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=1000007&projectid=3&pathid=25](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=25) |
| Status | New |

The size of the buffer used by LoadHeader in char, at line 190 of lua-5.2.4/src/lundump.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that LoadHeader passes to char, at line 190 of lua-5.2.4/src/lundump.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lundump.c | lua-5.2.4/src/lundump.c |
| Line | 195 | 195 |

| | | |
|---|---|---|
| Object | char | char |

Code Snippet
File Name lua-5.2.4/src/lundump.c
Method static void LoadHeader(LoadState* S)

```
....
195.    memcpy(s,h,sizeof(char));                    /* first char already
read */
```

**Buffer Overflow boundcpy WrongSizeParam\Path 2:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=26 |
| Status | New |

The size of the buffer used by *luaL_prepbuffsize in B, at line 437 of lua-5.2.4/src/lauxlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *luaL_prepbuffsize passes to B, at line 437 of lua-5.2.4/src/lauxlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 449 | 449 |
| Object | B | B |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API char *luaL_prepbuffsize (luaL_Buffer *B, size_t sz) {

```
....
449.      memcpy(newbuff, B->b, B->n * sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 3:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=27 |
| Status | New |

The size of the buffer used by *luaL_prepbuffsize in char, at line 437 of lua-5.2.4/src/lauxlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *luaL_prepbuffsize passes to char, at line 437 of lua-5.2.4/src/lauxlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 449 | 449 |
| Object | char | char |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API char *luaL_prepbuffsize (luaL_Buffer *B, size_t sz) {

```
....
449.        memcpy(newbuff, B->b, B->n * sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 4:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=28 |
| Status | New |

The size of the buffer used by luaL_addlstring in l, at line 459 of lua-5.2.4/src/lauxlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaL_addlstring passes to l, at line 459 of lua-5.2.4/src/lauxlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 461 | 461 |
| Object | l | l |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API void luaL_addlstring (luaL_Buffer *B, const char *s, size_t l) {

```
....
461.      memcpy(b, s, l * sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 5:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=29 |
| Status | New |

The size of the buffer used by luaL_addlstring in char, at line 459 of lua-5.2.4/src/lauxlib.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaL_addlstring passes to char, at line 459 of lua-5.2.4/src/lauxlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 461 | 461 |
| Object | char | char |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c

| | |
|---|---|
| Method | LUALIB_API void luaL_addlstring (luaL_Buffer *B, const char *s, size_t l) { |

```
....
461.    memcpy(b, s, l * sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 6:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=30](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=30) |
| Status | New |

The size of the buffer used by luaO_chunkid in l, at line 252 of lua-5.2.4/src/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to l, at line 252 of lua-5.2.4/src/lobject.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 256 | 256 |
| Object | l | l |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lobject.c |
| Method | void luaO_chunkid (char *out, const char *source, size_t bufflen) { |

```
....
256.        memcpy(out, source + 1, l * sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 7:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=31](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=31) |
| Status | New |

The size of the buffer used by luaO_chunkid in char, at line 252 of lua-5.2.4/src/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to char, at line 252 of lua-5.2.4/src/lobject.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 256 | 256 |
| Object | char | char |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lobject.c |
| Method | void luaO_chunkid (char *out, const char *source, size_t bufflen) { |

```
....
256.        memcpy(out, source + 1, l * sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 8:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=32 |
| Status | New |

The size of the buffer used by luaO_chunkid in l, at line 252 of lua-5.2.4/src/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to l, at line 252 of lua-5.2.4/src/lobject.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 264 | 264 |
| Object | l | l |

Code Snippet
File Name lua-5.2.4/src/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
264.        memcpy(out, source + 1, l * sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 9:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=33 |
| Status | New |

The size of the buffer used by luaO_chunkid in char, at line 252 of lua-5.2.4/src/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to char, at line 252 of lua-5.2.4/src/lobject.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 264 | 264 |
| Object | char | char |

Code Snippet
File Name lua-5.2.4/src/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
264.            memcpy(out, source + 1, l * sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 10:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=34 |
| Status | New |

The size of the buffer used by luaO_chunkid in bufflen, at line 252 of lua-5.2.4/src/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to bufflen, at line 252 of lua-5.2.4/src/lobject.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 268 | 268 |
| Object | bufflen | bufflen |

Code Snippet
File Name lua-5.2.4/src/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
268.            memcpy(out, source + 1 + l - bufflen, bufflen *
sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 11:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=35 |
| Status | New |

The size of the buffer used by luaO_chunkid in char, at line 252 of lua-5.2.4/src/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to char, at line 252 of lua-5.2.4/src/lobject.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 268 | 268 |
| Object | char | char |

Code Snippet
File Name lua-5.2.4/src/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
268.        memcpy(out, source + 1 + l - bufflen, bufflen *
sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 12:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=36 |
| Status | New |

The size of the buffer used by luaO_chunkid in char, at line 252 of lua-5.2.4/src/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to char, at line 252 of lua-5.2.4/src/lobject.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 284 | 284 |
| Object | char | char |

Code Snippet
File Name lua-5.2.4/src/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
284.       memcpy(out, POS, (LL(POS) + 1) * sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 13:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=37 |
| Status | New |

The size of the buffer used by *createstrobj in l, at line 98 of lua-5.2.4/src/lstring.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *createstrobj passes to l, at line 98 of lua-5.2.4/src/lstring.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstring.c | lua-5.2.4/src/lstring.c |
| Line | 107 | 107 |
| Object | l | l |

Code Snippet
File Name lua-5.2.4/src/lstring.c
Method static TString *createstrobj (lua_State *L, const char *str, size_t l,

```
....
107.     memcpy(ts+1, str, l*sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 14:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=38 |
| Status | New |

The size of the buffer used by *createstrobj in char, at line 98 of lua-5.2.4/src/lstring.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *createstrobj passes to char, at line 98 of lua-5.2.4/src/lstring.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstring.c | lua-5.2.4/src/lstring.c |
| Line | 107 | 107 |
| Object | char | char |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstring.c |
| Method | static TString *createstrobj (lua_State *L, const char *str, size_t l, |

```
....
107.     memcpy(ts+1, str, l*sizeof(char));
```

**Buffer Overflow boundcpy WrongSizeParam\Path 15:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=39 |
| Status | New |

The size of the buffer used by str_rep in l, at line 108 of lua-5.2.4/src/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 108 of lua-5.2.4/src/lstrlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 121 | 121 |
| Object | l | l |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static int str_rep (lua_State *L) { |

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

### Buffer Overflow boundcpy WrongSizeParam\Path 16:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=40 |
| Status | New |

The size of the buffer used by str_rep in char, at line 108 of lua-5.2.4/src/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 108 of lua-5.2.4/src/lstrlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 121 | 121 |
| Object | char | char |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static int str_rep (lua_State *L) { |

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

### Buffer Overflow boundcpy WrongSizeParam\Path 17:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=41 |
| Status | New |

The size of the buffer used by str_rep in lsep, at line 108 of lua-5.2.4/src/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 108 of lua-5.2.4/src/lstrlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 123 | 123 |
| Object | lsep | lsep |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static int str_rep (lua_State *L) { |

```
....
123.           memcpy(p, sep, lsep * sizeof(char)); p += lsep;
```

**Buffer Overflow boundcpy WrongSizeParam\Path 18:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=42](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=42) |
| Status | New |

The size of the buffer used by str_rep in char, at line 108 of lua-5.2.4/src/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 108 of lua-5.2.4/src/lstrlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 123 | 123 |
| Object | char | char |

Code Snippet
File Name     lua-5.2.4/src/lstrlib.c
Method     static int str_rep (lua_State *L) {

```
....
123.           memcpy(p, sep, lsep * sizeof(char)); p += lsep;
```

**Buffer Overflow boundcpy WrongSizeParam\Path 19:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=43](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=43) |
| Status | New |

The size of the buffer used by str_rep in l, at line 108 of lua-5.2.4/src/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 108 of lua-5.2.4/src/lstrlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 126 | 126 |
| Object | l | l |

Code Snippet
File Name     lua-5.2.4/src/lstrlib.c
Method     static int str_rep (lua_State *L) {

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

### Buffer Overflow boundcpy WrongSizeParam\Path 20:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=44 |
| Status | New |

The size of the buffer used by str_rep in char, at line 108 of lua-5.2.4/src/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 108 of lua-5.2.4/src/lstrlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 126 | 126 |
| Object | char | char |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static int str_rep (lua_State *L) {

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

### Buffer Overflow boundcpy WrongSizeParam\Path 21:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=45 |
| Status | New |

The size of the buffer used by *scanformat in char, at line 857 of lua-5.2.4/src/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 857 of lua-5.2.4/src/lstrlib.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 872 | 872 |
| Object | char | char |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static const char *scanformat (lua_State *L, const char *strfrmt, char *form) {

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

**Buffer Overflow boundcpy WrongSizeParam\Path 22:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=46](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=46) |
| Status | New |

The size of the buffer used by luaZ_read in m, at line 46 of lua-5.2.4/src/lzio.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaZ_read passes to m, at line 46 of lua-5.2.4/src/lzio.c, to overwrite the target buffer.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lzio.c | lua-5.2.4/src/lzio.c |
| Line | 58 | 58 |
| Object | m | m |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lzio.c |
| Method | size_t luaZ_read (ZIO *z, void *b, size_t n) { |

```
....
58.      memcpy(b, z->p, m);
```

## Buffer Overflow AddressOfLocalVarReturned

Query Path:
CPP\Cx\CPP Buffer Overflow\Buffer Overflow AddressOfLocalVarReturned Version:1

### Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
NIST SP 800-53: SC-5 Denial of Service Protection (P1)
OWASP Top 10 2017: A1-Injection

*Description*

**Buffer Overflow AddressOfLocalVarReturned\Path 1:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=20](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=20) |
| Status | New |

The pointer ts at lua-5.2.4/src/lstring.c in line 133 is being used after it has been freed.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstring.c | lua-5.2.4/src/lstring.c |
| Line | 146 | 146 |

| | | |
|---|---|---|
| Object | ts | ts |

Code Snippet
File Name lua-5.2.4/src/lstring.c
Method static TString *internshrstr (lua_State *L, const char *str, size_t l) {

```
....
146.         return ts;
```

**Buffer Overflow AddressOfLocalVarReturned\Path 2:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=21 |
| Status | New |

The pointer luaO_nilobject_ at lua-5.2.4/src/ltable.c in line 446 is being used after it has been freed.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ltable.c | lua-5.2.4/src/ltable.c |
| Line | 458 | 458 |
| Object | luaO_nilobject_ | luaO_nilobject_ |

Code Snippet
File Name lua-5.2.4/src/ltable.c
Method const TValue *luaH_getint (Table *t, int key) {

```
....
458.       return luaO_nilobject;
```

**Buffer Overflow AddressOfLocalVarReturned\Path 3:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=22 |
| Status | New |

The pointer luaO_nilobject_ at lua-5.2.4/src/ltable.c in line 466 is being used after it has been freed.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ltable.c | lua-5.2.4/src/ltable.c |
| Line | 474 | 474 |
| Object | luaO_nilobject_ | luaO_nilobject_ |

Code Snippet
File Name lua-5.2.4/src/ltable.c

Method const TValue *luaH_getstr (Table *t, TString *key) {

```
....
474.     return luaO_nilobject;
```

### Buffer Overflow AddressOfLocalVarReturned\Path 4:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=23 |
| Status | New |

The pointer luaO_nilobject_ at lua-5.2.4/src/ltable.c in line 481 is being used after it has been freed.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ltable.c | lua-5.2.4/src/ltable.c |
| Line | 484 | 484 |
| Object | luaO_nilobject_ | luaO_nilobject_ |

Code Snippet
File Name lua-5.2.4/src/ltable.c
Method const TValue *luaH_get (Table *t, const TValue *key) {

```
....
484.      case LUA_TNIL: return luaO_nilobject;
```

### Buffer Overflow AddressOfLocalVarReturned\Path 5:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=24 |
| Status | New |

The pointer luaO_nilobject_ at lua-5.2.4/src/ltable.c in line 481 is being used after it has been freed.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ltable.c | lua-5.2.4/src/ltable.c |
| Line | 500 | 500 |
| Object | luaO_nilobject_ | luaO_nilobject_ |

Code Snippet
File Name lua-5.2.4/src/ltable.c
Method const TValue *luaH_get (Table *t, const TValue *key) {

```
....
500.       return luaO_nilobject;
```

# Use of Zero Initialized Pointer

Query Path:
CPP\Cx\CPP Medium Threat\Use of Zero Initialized Pointer Version:1

## Categories

NIST SP 800-53: SC-5 Denial of Service Protection (P1)

Description

### Use of Zero Initialized Pointer\Path 1:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=106 |
| Status | New |

The variable declared in varname at lua-5.2.4/src/lparser.c in line 165 is not initialized when it is used by varname at lua-5.2.4/src/lparser.c in line 165.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lparser.c | lua-5.2.4/src/lparser.c |
| Line | 171 | 172 |
| Object | varname | varname |

Code Snippet
File Name lua-5.2.4/src/lparser.c
Method static int registerlocalvar (LexState *ls, TString *varname) {

```
....
171.    while (oldsize < f->sizelocvars) f->locvars[oldsize++].varname =
NULL;
172.    f->locvars[fs->nlocvars].varname = varname;
```

### Use of Zero Initialized Pointer\Path 2:

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=107 |
| Status | New |

The variable declared in name at lua-5.2.4/src/lparser.c in line 231 is not initialized when it is used by name at lua-5.2.4/src/lparser.c in line 231.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lparser.c | lua-5.2.4/src/lparser.c |
| Line | 237 | 240 |
| Object | name | name |

| | |
|---|---|
| Code Snippet | |
| File Name | lua-5.2.4/src/lparser.c |
| Method | static int newupvalue (FuncState *fs, TString *name, expdesc *v) { |

```
....
237.     while (oldsize < f->sizeupvalues) f->upvalues[oldsize++].name =
NULL;
....
240.     f->upvalues[fs->nups].name = name;
```

**Use of Zero Initialized Pointer\Path 3:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=108 |
| Status | New |

The variable declared in prev at lua-5.2.4/src/lparser.c in line 1480 is not initialized when it is used by prev at lua-5.2.4/src/lparser.c in line 1136.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lparser.c | lua-5.2.4/src/lparser.c |
| Line | 1486 | 1141 |
| Object | prev | prev |

| | |
|---|---|
| Code Snippet | |
| File Name | lua-5.2.4/src/lparser.c |
| Method | static void exprstat (LexState *ls) { |

```
....
1486.       v.prev = NULL;
```

| | |
|---|---|
| File Name | lua-5.2.4/src/lparser.c |
| Method | static void assignment (LexState *ls, struct LHS_assign *lh, int nvars) { |

```
....
1141.       nv.prev = lh;
```

## Integer Overflow

Query Path:
CPP\Cx\CPP Integer Overflow\Integer Overflow Version:0

### Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)

*Description*

**Integer Overflow\Path 1:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=52](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=52) |
| Status | New |

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 133 of lua-5.2.4/src/lstrlib.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 142 | 142 |
| Object | AssignExpr | AssignExpr |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static int str_byte (lua_State *L) {

```
....
142.    n = (int)(pose -  posi + 1);
```

**Integer Overflow\Path 2:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=53](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=53) |
| Status | New |

A variable of a larger data type, h, is being assigned to a smaller data type, in 51 of lua-5.2.4/src/lstring.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstring.c | lua-5.2.4/src/lstring.c |
| Line | 52 | 52 |
| Object | h | h |

Code Snippet
File Name lua-5.2.4/src/lstring.c
Method unsigned int luaS_hash (const char *str, size_t l, unsigned int seed) {

```
....
52.    unsigned int h = seed ^ cast(unsigned int, l);
```

## Wrong Size t Allocation

Query Path:
CPP\Cx\CPP Integer Overflow\Wrong Size t Allocation Version:0

*Description*

**Wrong Size t Allocation\Path 1:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=47 |
| Status | New |

The function nsize in lua-5.2.4/src/lauxlib.c at line 919 assigns an incorrectly calculated size to a buffer, resulting in a mismatch between the value being written and the size of the buffer it is being written into.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 926 | 926 |
| Object | nsize | nsize |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method static void *l_alloc (void *ud, void *ptr, size_t osize, size_t nsize) {

```
....
926.      return realloc(ptr, nsize);
```

## 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=1000007&projectid=3&pathid=109 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ldblib.c | lua-5.2.4/src/ldblib.c |
| Line | 358 | 358 |
| Object | fgets | fgets |

Code Snippet
File Name lua-5.2.4/src/ldblib.c
Method static int db_debug (lua_State *L) {

```
....
358.        if (fgets(buffer, sizeof(buffer), stdin) == 0 ||
```

**Improper Resource Access Authorization\Path 2:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=110 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 377 | 377 |
| Object | fgets | fgets |

Code Snippet
File Name    lua-5.2.4/src/liolib.c
Method    static int read_line (lua_State *L, FILE *f, int chop) {

```
....
377.        if (fgets(p, LUAL_BUFFERSIZE, f) == NULL) {  /* eof? */
```

**Improper Resource Access Authorization\Path 3:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=111 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 352 | 352 |
| Object | fscanf | fscanf |

Code Snippet
File Name    lua-5.2.4/src/liolib.c
Method    static int read_number (lua_State *L, FILE *f) {

```
....
352.    if (fscanf(f, LUA_NUMBER_SCAN, &d) == 1) {
```

**Improper Resource Access Authorization\Path 4:**

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

| | |
|---|---|
| | BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=112 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ldblib.c | lua-5.2.4/src/ldblib.c |
| Line | 358 | 358 |
| Object | buffer | buffer |

Code Snippet
File Name lua-5.2.4/src/ldblib.c
Method static int db_debug (lua_State *L) {

```
....
358.      if (fgets(buffer, sizeof(buffer), stdin) == 0 ||
```

**Improper Resource Access Authorization\Path 5:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=113 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 377 | 377 |
| Object | p | p |

Code Snippet
File Name lua-5.2.4/src/liolib.c
Method static int read_line (lua_State *L, FILE *f, int chop) {

```
....
377.      if (fgets(p, LUAL_BUFFERSIZE, f) == NULL) {  /* eof? */
```

**Improper Resource Access Authorization\Path 6:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=114 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 581 | 581 |

| | | |
|---|---|---|
| Object | buff | buff |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method static const char *getF (lua_State *L, void *ud, size_t *size) {

```
....
581.      *size = fread(lf->buff, 1, sizeof(lf->buff), lf->f);  /* read
block */
```

**Improper Resource Access Authorization\Path 7:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=115 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 401 | 401 |
| Object | p | p |

Code Snippet
File Name lua-5.2.4/src/liolib.c
Method static void read_all (lua_State *L, FILE *f) {

```
....
401.      size_t nr = fread(p, sizeof(char), rlen, f);
```

**Improper Resource Access Authorization\Path 8:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=116 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 417 | 417 |
| Object | p | p |

Code Snippet
File Name lua-5.2.4/src/liolib.c
Method static int read_chars (lua_State *L, FILE *f, size_t n) {

```
....
417.    nr = fread(p, sizeof(char), n, f);  /* try to read 'n' chars */
```

### Improper Resource Access Authorization\Path 9:

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=117 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 352 | 352 |
| Object | Address | Address |

Code Snippet
File Name lua-5.2.4/src/liolib.c
Method static int read_number (lua_State *L, FILE *f) {

```
....
352.    if (fscanf(f, LUA_NUMBER_SCAN, &d) == 1) {
```

### Improper Resource Access Authorization\Path 10:

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=118 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 521 | 521 |
| Object | fprintf | fprintf |

Code Snippet
File Name lua-5.2.4/src/liolib.c
Method static int g_write (lua_State *L, FILE *f, int arg) {

```
....
521.            fprintf(f, LUA_NUMBER_FMT, lua_tonumber(L, arg)) > 0;
```

### Improper Resource Access Authorization\Path 11:

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

| | |
|---|---|
| | BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=119 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/luac.c | lua-5.2.4/src/luac.c |
| Line | 37 | 37 |
| Object | fprintf | fprintf |

Code Snippet
File Name lua-5.2.4/src/luac.c
Method static void fatal(const char* message)

```
....
37.    fprintf(stderr,"%s: %s\n",progname,message);
```

**Improper Resource Access Authorization\Path 12:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=120 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/luac.c | lua-5.2.4/src/luac.c |
| Line | 43 | 43 |
| Object | fprintf | fprintf |

Code Snippet
File Name lua-5.2.4/src/luac.c
Method static void cannot(const char* what)

```
....
43.    fprintf(stderr,"%s: cannot %s %s:
%s\n",progname,what,output,strerror(errno));
```

**Improper Resource Access Authorization\Path 13:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=121 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/luac.c | lua-5.2.4/src/luac.c |

| | | |
|---|---|---|
| Line | 50 | 50 |
| Object | fprintf | fprintf |

Code Snippet
File Name lua-5.2.4/src/luac.c
Method static void usage(const char* message)

```
....
50.     fprintf(stderr,"%s: unrecognized option " LUA_QS
"\n",progname,message);
```

**Improper Resource Access Authorization\Path 14:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=122 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/luac.c | lua-5.2.4/src/luac.c |
| Line | 52 | 52 |
| Object | fprintf | fprintf |

Code Snippet
File Name lua-5.2.4/src/luac.c
Method static void usage(const char* message)

```
....
52.     fprintf(stderr,"%s: %s\n",progname,message);
```

**Improper Resource Access Authorization\Path 15:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=123 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/luac.c | lua-5.2.4/src/luac.c |
| Line | 53 | 53 |
| Object | fprintf | fprintf |

Code Snippet
File Name lua-5.2.4/src/luac.c
Method static void usage(const char* message)

```
....
53.    fprintf(stderr,
```

### Improper Resource Access Authorization\Path 16:

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=124 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lbaselib.c | lua-5.2.4/src/lbaselib.c |
| Line | 37 | 37 |
| Object | fwrite | fwrite |

Code Snippet
File Name lua-5.2.4/src/lbaselib.c
Method static int luaB_print (lua_State *L) {

```
....
37.       if (i>1) luai_writestring("\t", 1);
```

### Improper Resource Access Authorization\Path 17:

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=125 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lbaselib.c | lua-5.2.4/src/lbaselib.c |
| Line | 38 | 38 |
| Object | fwrite | fwrite |

Code Snippet
File Name lua-5.2.4/src/lbaselib.c
Method static int luaB_print (lua_State *L) {

```
....
38.       luai_writestring(s, l);
```

### Improper Resource Access Authorization\Path 18:

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

| | |
|---|---|
| | BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=126 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 526 | 526 |
| Object | fwrite | fwrite |

Code Snippet
File Name lua-5.2.4/src/liolib.c
Method static int g_write (lua_State *L, FILE *f, int arg) {

```
....
526.         status = status && (fwrite(s, sizeof(char), l, f) == l);
```

**Improper Resource Access Authorization\Path 19:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=127 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lua.c | lua-5.2.4/src/lua.c |
| Line | 187 | 187 |
| Object | fwrite | fwrite |

Code Snippet
File Name lua-5.2.4/src/lua.c
Method static void print_version (void) {

```
....
187.    luai_writestring(LUA_COPYRIGHT, strlen(LUA_COPYRIGHT));
```

**Improper Resource Access Authorization\Path 20:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=128 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/luac.c | lua-5.2.4/src/luac.c |
| Line | 159 | 159 |

| | | |
|---|---|---|
| Object | fwrite | fwrite |

Code Snippet
File Name lua-5.2.4/src/luac.c
Method static int writer(lua_State* L, const void* p, size_t size, void* u)

```
....
159.    return (fwrite(p,size,1,(FILE*)u)!=1) && (size!=0);
```

## Unchecked Return Value

Query Path:
CPP\Cx\CPP Low Visibility\Unchecked Return Value Version:1

### Categories

NIST SP 800-53: SI-11 Error Handling (P2)

*Description*

**Unchecked Return Value\Path 1:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=4 |
| Status | New |

The *l_alloc method calls the realloc function, at line 919 of lua-5.2.4/src/lauxlib.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 926 | 926 |
| Object | realloc | realloc |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method static void *l_alloc (void *ud, void *ptr, size_t osize, size_t nsize) {

```
....
926.      return realloc(ptr, nsize);
```

**Unchecked Return Value\Path 2:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=5 |
| Status | New |

The os_remove method calls the remove function, at line 92 of lua-5.2.4/src/loslib.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/loslib.c | lua-5.2.4/src/loslib.c |
| Line | 94 | 94 |
| Object | remove | remove |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/loslib.c |
| Method | static int os_remove (lua_State *L) { |

```
....
94.    return luaL_fileresult(L, remove(filename) == 0, filename);
```

**Unchecked Return Value\Path 3:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=6 |
| Status | New |

The addquoted method calls the sprintf function, at line 833 of lua-5.2.4/src/lstrlib.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 845 | 845 |
| Object | sprintf | sprintf |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lstrlib.c |
| Method | static void addquoted (lua_State *L, luaL_Buffer *b, int arg) { |

```
....
845.          sprintf(buff, "\\%d", (int)uchar(*s));
```

**Unchecked Return Value\Path 4:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=7 |
| Status | New |

The addquoted method calls the sprintf function, at line 833 of lua-5.2.4/src/lstrlib.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 847 | 847 |
| Object | sprintf | sprintf |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static void addquoted (lua_State *L, luaL_Buffer *b, int arg) {

```
....
847.            sprintf(buff, "\\%03d", (int)uchar(*s));
```

**Unchecked Return Value\Path 5:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=8 |
| Status | New |

The str_format method calls the nb function, at line 892 of lua-5.2.4/src/lstrlib.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 914 | 914 |
| Object | nb | nb |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static int str_format (lua_State *L) {

```
....
914.              nb = sprintf(buff, form, luaL_checkint(L, arg));
```

**Unchecked Return Value\Path 6:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=9 |
| Status | New |

The str_format method calls the nb function, at line 892 of lua-5.2.4/src/lstrlib.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 924 | 924 |
| Object | nb | nb |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static int str_format (lua_State *L) {

```
....
924.              nb = sprintf(buff, form, ni);
```

**Unchecked Return Value\Path 7:**

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=10
Status New

The str_format method calls the nb function, at line 892 of lua-5.2.4/src/lstrlib.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 934 | 934 |
| Object | nb | nb |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static int str_format (lua_State *L) {

```
....
934.              nb = sprintf(buff, form, ni);
```

**Unchecked Return Value\Path 8:**

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=11
Status New

The str_format method calls the nb function, at line 892 of lua-5.2.4/src/lstrlib.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |

| | | |
|---|---|---|
| Line | 943 | 943 |
| Object | nb | nb |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static int str_format (lua_State *L) {

```
....
943.            nb = sprintf(buff, form,
(LUA_FLTFRM_T)luaL_checknumber(L, arg));
```

**Unchecked Return Value\Path 9:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=12 |
| Status | New |

The str_format method calls the nb function, at line 892 of lua-5.2.4/src/lstrlib.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstrlib.c | lua-5.2.4/src/lstrlib.c |
| Line | 960 | 960 |
| Object | nb | nb |

Code Snippet
File Name lua-5.2.4/src/lstrlib.c
Method static int str_format (lua_State *L) {

```
....
960.              nb = sprintf(buff, form, s);
```

## 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=1000007&projectid=3&pathid=13 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ldump.c | lua-5.2.4/src/ldump.c |

| | | |
|---|---|---|
| Line | 45 | 47 |
| Object | x | sizeof |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/ldump.c |
| Method | static void DumpInt(int x, DumpState* D) |

```
....
45.  static void DumpInt(int x, DumpState* D)
....
47.   DumpVar(x,D);
```

### Use of Sizeof On a Pointer Type\Path 2:

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=14 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ldump.c | lua-5.2.4/src/ldump.c |
| Line | 50 | 52 |
| Object | x | sizeof |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/ldump.c |
| Method | static void DumpNumber(lua_Number x, DumpState* D) |

```
....
50.  static void DumpNumber(lua_Number x, DumpState* D)
....
52.   DumpVar(x,D);
```

### Use of Sizeof On a Pointer Type\Path 3:

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=15 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lgc.c | lua-5.2.4/src/lgc.c |
| Line | 471 | 471 |
| Object | sizeof | sizeof |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lgc.c |
| Method | static int traverseproto (global_State *g, Proto *f) { |

```
....
471.                             sizeof(Proto *) * f->sizep +
```

**Use of Sizeof On a Pointer Type\Path 4:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=16 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lgc.c | lua-5.2.4/src/lgc.c |
| Line | 1045 | 1045 |
| Object | sizeof | sizeof |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lgc.c |
| Method | static lu_mem singlestep (lua_State *L) { |

```
....
1045.        g->GCmemtrav = g->strt.size * sizeof(GCObject*);
```

**Use of Sizeof On a Pointer Type\Path 5:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=17 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lobject.c | lua-5.2.4/src/lobject.c |
| Line | 208 | 208 |
| Object | sizeof | sizeof |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lobject.c |
| Method | const char *luaO_pushvfstring (lua_State *L, const char *fmt, va_list argp) { |

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

## TOCTOU

Query Path:
CPP\Cx\CPP Low Visibility\TOCTOU Version:1

Description

**TOCTOU\Path 1:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=131 |
| Status | New |

The luaL_loadfilex method in lua-5.2.4/src/lauxlib.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lauxlib.c | lua-5.2.4/src/lauxlib.c |
| Line | 642 | 642 |
| Object | fopen | fopen |

Code Snippet
File Name lua-5.2.4/src/lauxlib.c
Method LUALIB_API int luaL_loadfilex (lua_State *L, const char *filename,

```
....
642.      lf.f = fopen(filename, "r");
```

**TOCTOU\Path 2:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=132 |
| Status | New |

The opencheck method in lua-5.2.4/src/liolib.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 220 | 220 |
| Object | fopen | fopen |

Code Snippet
File Name lua-5.2.4/src/liolib.c
Method static void opencheck (lua_State *L, const char *fname, const char *mode) {

```
....
220.    p->f = fopen(fname, mode);
```

**TOCTOU\Path 3:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=133 |
| Status | New |

The io_open method in lua-5.2.4/src/liolib.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 232 | 232 |
| Object | fopen | fopen |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/liolib.c |
| Method | static int io_open (lua_State *L) { |

```
....
232.     p->f = fopen(filename, mode);
```

**TOCTOU\Path 4:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=134 |
| Status | New |

The readable method in lua-5.2.4/src/loadlib.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/loadlib.c | lua-5.2.4/src/loadlib.c |
| Line | 331 | 331 |
| Object | fopen | fopen |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/loadlib.c |
| Method | static int readable (const char *filename) { |

```
....
331.     FILE *f = fopen(filename, "r");   /* try to open file */
```

**TOCTOU\Path 5:**

| | |
|---|---|
| Severity | Low |

| | |
|---|---|
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=135 |
| Status | New |

The pmain method in lua-5.2.4/src/luac.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/luac.c | lua-5.2.4/src/luac.c |
| Line | 178 | 178 |
| Object | fopen | fopen |

Code Snippet
File Name lua-5.2.4/src/luac.c
Method static int pmain(lua_State* L)

```
....
178.     FILE* D= (output==NULL) ? stdout : fopen(output,"wb");
```

## 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=1000007&projectid=3&pathid=58 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ldblib.c | lua-5.2.4/src/ldblib.c |
| Line | 299 | 299 |
| Object | i | i |

Code Snippet
File Name lua-5.2.4/src/ldblib.c
Method static char *unmakemask (int mask, char *smask) {

```
....
299.     smask[i] = '\0';
```

**Unchecked Array Index\Path 2:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=59 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lfunc.c | lua-5.2.4/src/lfunc.c |
| Line | 34 | 34 |
| Object | n | n |

Code Snippet
File Name lua-5.2.4/src/lfunc.c
Method Closure *luaF_newLclosure (lua_State *L, int n) {

```
....
34.     while (n--) c->l.upvals[n] = NULL;
```

**Unchecked Array Index\Path 3:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=60 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstring.c | lua-5.2.4/src/lstring.c |
| Line | 81 | 81 |
| Object | h | h |

Code Snippet
File Name lua-5.2.4/src/lstring.c
Method void luaS_resize (lua_State *L, int newsize) {

```
....
81.         tb->hash[h] = p;
```

**Unchecked Array Index\Path 4:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=61 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lstring.c | lua-5.2.4/src/lstring.c |
| Line | 108 | 108 |
| Object | l | l |

Code Snippet
File Name	lua-5.2.4/src/lstring.c
Method	static TString *createstrobj (lua_State *L, const char *str, size_t l,

```
....
108.     ((char *)(ts+1))[l] = '\0';  /* ending 0 */
```

## Use of Insufficiently Random Values

Query Path:
CPP\Cx\CPP Low Visibility\Use of Insufficiently Random Values Version:0

### Categories

FISMA 2014: Media Protection
NIST SP 800-53: SC-28 Protection of Information at Rest (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

*Description*

**Use of Insufficiently Random Values\Path 1:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=1 |
| Status | New |

Method math_random at line 198 of lua-5.2.4/src/lmathlib.c uses a weak method rand to produce random values. These values might be used for secret values, personal identifiers or cryptographic input, allowing an attacker to guess the value.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lmathlib.c | lua-5.2.4/src/lmathlib.c |
| Line | 201 | 201 |
| Object | rand | rand |

Code Snippet
File Name	lua-5.2.4/src/lmathlib.c
Method	static int math_random (lua_State *L) {

```
....
201.     lua_Number r = (lua_Number)(rand()%RAND_MAX) /
(lua_Number)RAND_MAX;
```

**Use of Insufficiently Random Values\Path 2:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |

| | |
|---|---|
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=2 |
| Status | New |

Method math_randomseed at line 226 of lua-5.2.4/src/lmathlib.c uses a weak method rand to produce random values. These values might be used for secret values, personal identifiers or cryptographic input, allowing an attacker to guess the value.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lmathlib.c | lua-5.2.4/src/lmathlib.c |
| Line | 228 | 228 |
| Object | rand | rand |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lmathlib.c |
| Method | static int math_randomseed (lua_State *L) { |

```
....
228.     (void)rand(); /* discard first value to avoid undesirable
correlations */
```

**Use of Insufficiently Random Values\Path 3:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=3 |
| Status | New |

Method math_randomseed at line 226 of lua-5.2.4/src/lmathlib.c uses a weak method srand to produce random values. These values might be used for secret values, personal identifiers or cryptographic input, allowing an attacker to guess the value.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lmathlib.c | lua-5.2.4/src/lmathlib.c |
| Line | 227 | 227 |
| Object | srand | srand |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/lmathlib.c |
| Method | static int math_randomseed (lua_State *L) { |

```
....
227.     srand(luaL_checkunsigned(L, 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=1000007&projectid=3&pathid=48 |
| Status | New |

The variable declared in null at lua-5.2.4/src/lgc.c in line 434 is not initialized when it is used by u at lua-5.2.4/src/lgc.c in line 434.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lgc.c | lua-5.2.4/src/lgc.c |
| Line | 436 | 438 |
| Object | null | u |

Code Snippet
File Name lua-5.2.4/src/lgc.c
Method static lu_mem traversetable (global_State *g, Table *h) {

```
....
436.    const TValue *mode = gfasttm(g, h->metatable, TM_MODE);
....
438.    if (mode && ttisstring(mode) &&  /* is there a weak mode? */
```

**NULL Pointer Dereference\Path 2:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=49 |
| Status | New |

The variable declared in null at lua-5.2.4/src/lgc.c in line 434 is not initialized when it is used by u at lua-5.2.4/src/lgc.c in line 434.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/lgc.c | lua-5.2.4/src/lgc.c |
| Line | 436 | 438 |
| Object | null | u |

Code Snippet
File Name lua-5.2.4/src/lgc.c
Method static lu_mem traversetable (global_State *g, Table *h) {

```
....
436.      const TValue *mode = gfasttm(g, h->metatable, TM_MODE);
....
438.      if (mode && ttisstring(mode) &&  /* is there a weak mode? */
```

**NULL Pointer Dereference\Path 3:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=50](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=50) |
| Status | New |

The variable declared in 0 at lua-5.2.4/src/ltable.c in line 368 is not initialized when it is used by t at lua-5.2.4/src/ltable.c in line 368.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ltable.c | lua-5.2.4/src/ltable.c |
| Line | 371 | 371 |
| Object | 0 | t |

Code Snippet
File Name lua-5.2.4/src/ltable.c
Method Table *luaH_new (lua_State *L) {

```
....
371.      t->flags = cast_byte(~0);
```

## Potential Off by One Error in Loops

Query Path:
CPP\Cx\CPP Heuristic\Potential Off by One Error in Loops Version:1

### Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection
NIST SP 800-53: SI-16 Memory Protection (P1)
OWASP Top 10 2017: A1-Injection

*Description*

**Potential Off by One Error in Loops\Path 1:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | [http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=18](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=18) |
| Status | New |

The buffer allocated by <= in lua-5.2.4/src/ltable.c at line 229 does not correctly account for the actual size of the value, resulting in an incorrect allocation that is off by one.

| | Source | Destination |
|---|---|---|

| | | |
|---|---|---|
| File | lua-5.2.4/src/ltable.c | lua-5.2.4/src/ltable.c |
| Line | 234 | 234 |
| Object | <= | <= |

Code Snippet
File Name lua-5.2.4/src/ltable.c
Method static int numusearray (const Table *t, int *nums) {

```
....
234.    for (lg=0, ttlg=1; lg<=MAXBITS; lg++, ttlg*=2) {  /* for each
slice */
```

**Potential Off by One Error in Loops\Path 2:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=19 |
| Status | New |

The buffer allocated by <= in lua-5.2.4/src/ltable.c at line 343 does not correctly account for the actual size of the value, resulting in an incorrect allocation that is off by one.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/ltable.c | lua-5.2.4/src/ltable.c |
| Line | 348 | 348 |
| Object | <= | <= |

Code Snippet
File Name lua-5.2.4/src/ltable.c
Method static void rehash (lua_State *L, Table *t, const TValue *ek) {

```
....
348.    for (i=0; i<=MAXBITS; i++) nums[i] = 0;  /* reset counts */
```

## Insecure Temporary File

Query Path:
CPP\Cx\CPP Low Visibility\Insecure Temporary File Version:0

### Categories

NIST SP 800-53: SC-4 Information in Shared Resources (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

*Description*

**Insecure Temporary File\Path 1:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=54 |

| | |
|---|---|
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/loslib.c | lua-5.2.4/src/loslib.c |
| Line | 108 | 108 |
| Object | mkstemp | mkstemp |

Code Snippet
File Name lua-5.2.4/src/loslib.c
Method static int os_tmpname (lua_State *L) {

```
....
108.     lua_tmpnam(buff, err);
```

**Insecure Temporary File\Path 2:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=55 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/liolib.c | lua-5.2.4/src/liolib.c |
| Line | 258 | 258 |
| Object | tmpfile | tmpfile |

Code Snippet
File Name lua-5.2.4/src/liolib.c
Method static int io_tmpfile (lua_State *L) {

```
....
258.     p->f = tmpfile();
```

## Sizeof Pointer Argument

Query Path:
CPP\Cx\CPP Low Visibility\Sizeof Pointer Argument Version:0

*Description*

**Sizeof Pointer Argument\Path 1:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=56 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/loslib.c | lua-5.2.4/src/loslib.c |

| | | |
|---|---|---|
| Line | 174 | 174 |
| Object | options | sizeof |

Code Snippet
File Name lua-5.2.4/src/loslib.c
Method static const char *checkoption (lua_State *L, const char *conv, char *buff) {

```
....
174.     for (i = 0; i < sizeof(options)/sizeof(options[0]); i += 2) {
```

**Sizeof Pointer Argument\Path 2:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=57 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/loslib.c | lua-5.2.4/src/loslib.c |
| Line | 174 | 174 |
| Object | options | sizeof |

Code Snippet
File Name lua-5.2.4/src/loslib.c
Method static const char *checkoption (lua_State *L, const char *conv, char *buff) {

```
....
174.     for (i = 0; i < sizeof(options)/sizeof(options[0]); i += 2) {
```

## Client Insufficient ClickJacking Protection

Query Path:
JavaScript\Cx\JavaScript Low Visibility\Client Insufficient ClickJacking Protection Version:1

### Categories

FISMA 2014: Configuration Management
NIST SP 800-53: SC-8 Transmission Confidentiality and Integrity (P1)

*Description*

**Client Insufficient ClickJacking Protection\Path 1:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=51 |
| Status | New |

The application does not protect the web page lua-5.2.4/doc/contents.html from clickjacking attacks in legacy browsers, by using framebusting scripts.

|  | Source | Destination |
|---|---|---|
| File | lua-5.2.4/doc/contents.html | lua-5.2.4/doc/contents.html |
| Line | 1 | 1 |
| Object | CxJSNS_968963717 | CxJSNS_968963717 |

Code Snippet
File Name lua-5.2.4/doc/contents.html
Method <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

```
....
1.  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

## Incorrect Permission Assignment For Critical Resources

Query Path:
CPP\Cx\CPP Low Visibility\Incorrect Permission Assignment For Critical Resources Version:1

### Categories

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

### Description

**Incorrect Permission Assignment For Critical Resources\Path 1:**

|  |  |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=129 |
| Status | New |

|  | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/loadlib.c | lua-5.2.4/src/loadlib.c |
| Line | 331 | 331 |
| Object | f | f |

Code Snippet
File Name lua-5.2.4/src/loadlib.c
Method static int readable (const char *filename) {

```
....
331.    FILE *f = fopen(filename, "r");  /* try to open file */
```

## Exposure of System Data to Unauthorized Control Sphere

Query Path:
CPP\Cx\CPP Low Visibility\Exposure of System Data to Unauthorized Control Sphere Version:1

### Categories

FISMA 2014: Configuration Management
NIST SP 800-53: AC-3 Access Enforcement (P1)

Description

**Exposure of System Data to Unauthorized Control Sphere\Path 1:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1000007&projectid=3&pathid=130 |
| Status | New |

The system data read by cannot in the file lua-5.2.4/src/luac.c at line 41 is potentially exposed by cannot found in lua-5.2.4/src/luac.c at line 41.

| | Source | Destination |
|---|---|---|
| File | lua-5.2.4/src/luac.c | lua-5.2.4/src/luac.c |
| Line | 43 | 43 |
| Object | errno | fprintf |

Code Snippet

| | |
|---|---|
| File Name | lua-5.2.4/src/luac.c |
| Method | static void cannot(const char* what) |

```
....
43.    fprintf(stderr,"%s: cannot %s %s:
%s\n",progname,what,output,strerror(errno));
```

# Buffer Overflow IndexFromInput

## 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

# Buffer Overflow AddressOfLocalVarReturned

## Risk

### What might happen

A use after free error will cause code to use an area of memory previously assigned with a specific value, which has since been freed and may have been overwritten by another value. This error will likely cause unexpected behavior, memory corruption and crash errors. In some cases where the freed and used section of memory is used to determine execution flow, and the error can be induced by an attacker, this may result in execution of malicious code.

## Cause

### How does it happen

Pointers to variables allow code to have an address with a set size to a dynamically allocated variable. Eventually, the pointer's destination may become free - either explicitly in code, such as when programmatically freeing this variable, or implicitly, such as when a local variable is returned - once it is returned, the variable's scope is released. Once freed, this memory will be re-used by the application, overwritten with new data. At this point, dereferencing this pointer will potentially resolve newly written and unexpected data.

## General Recommendations

### How to avoid it

- Do not return local variables or pointers
- Review code to ensure no flow allows use of a pointer after it has been explicitly freed

## Source Code Examples

### CPP

### Use of Variable after It was Freed

```
free(input);
printf("%s", input);
```

### Use of Pointer to Local Variable That Was Freed On Return

```
int* func1()
{
	int i;
	i = 1;
	return &i;
}

void func2()
```

```
{
      int j;
      j = 5;
}

//..
      int * i = func1();
      printf("%d\r\n", *i); // Output could be 1 or Segmentation Fault
      func2();
      printf("%d\r\n", *i); // Output is 5, which is j's value, as func2() overwrote data in
the stack
//..
```

# 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));
    }
}
```

# Wrong Size t Allocation

## Risk

### What might happen

Incorrect allocation of memory may result in unexpected behavior by either overwriting sections of memory with unexpected values. Under certain conditions where both an incorrect allocation of memory and the values being written can be controlled by an attacker, such an issue may result in execution of malicious code.

## Cause

### How does it happen

Some memory allocation functions require a size value to be provided as a parameter. The allocated size should be derived from the provided value, by providing the length value of the intended source, multiplied by the size of that length. Failure to perform the correct arithmetic to obtain the exact size of the value will likely result in the source overflowing its destination.

## General Recommendations

### How to avoid it

- Always perform the correct arithmetic to determine size.
- Specifically for memory allocation, calculate the allocation size from the allocation source:
  - Derive the size value from the length of intended source to determine the amount of units to be processed.
  - Always programmatically consider the size of the each unit and their conversion to memory units - for example, by using sizeof() on the unit's type.
  - Memory allocation should be a multiplication of the amount of units being written, times the size of each unit.

## Source Code Examples

### CPP

### Allocating and Assigning Memory without Sizeof Arithmetic

```
int *ptr;
ptr = (int*)malloc(5);
for (int i = 0; i < 5; i++)
{
      ptr[i] = i * 2 + 1;
}
```

### Allocating and Assigning Memory with Sizeof Arithmetic

```
int *ptr;
ptr = (int*)malloc(5 * sizeof(int));
```

```
for (int i = 0; i < 5; i++)
{
      ptr[i] = i * 2 + 1;
}
```

### Incorrect Arithmetic of Multi-Byte String Allocation

```
wchar_t * dest;
dest = (wchar_t *)malloc(wcslen(source) + 1); // Would not crash for a short "source"
wcscpy((wchar_t *)dest, source);
wprintf(L"Dest: %s\r\n", dest);
```

### Correct Arithmetic of Multi-Byte String Allocation

```
wchar_t * dest;
dest = (wchar_t *)malloc((wcslen(source) + 1) * sizeof(wchar_t));
wcscpy((wchar_t *)dest, source);
wprintf(L"Dest: %s\r\n", dest);
```

# Integer Overflow

## Risk

### What might happen

Assigning large data types into smaller data types, without proper checks and explicit casting, will lead to undefined behavior and unintentional effects, such as data corruption (e.g. value wraparound, wherein maximum values become minimum values); system crashes; infinite loops; logic errors, such as bypassing of security mechanisms; or even buffer overflows leading to arbitrary code execution.

## Cause

### How does it happen

This flaw can occur when implicitly casting numerical data types of a larger size, into a variable with a data type of a smaller size. This forces the program to discard some bits of information from the number. Depending on how the numerical data types are stored in memory, this is often the bits with the highest value, causing substantial corruption of the stored number. Alternatively, the sign bit of a signed integer could be lost, completely reversing the intention of the number.

## General Recommendations

### How to avoid it

- Avoid casting larger data types to smaller types.
- Prefer promoting the target variable to a large enough data type.
- If downcasting is necessary, always check that values are valid and in range of the target type, before casting

## Source Code Examples

### CPP

### Unsafe Downsize Casting

```
int unsafe_addition(short op1, int op2) {

    // op2 gets forced from int into a short
    short total = op1 + op2;

    return total;
}
```

### Safer Use of Proper Data Types

```
int safe_addition(short op1, int op2) {

    // total variable is of type int, the largest type that is needed
    int total = 0;

    // check if total will overflow available integer size
    if (INT_MAX - abs(op2) > op1)
    {
        total = op1 + op2;
    }
    else
    {
        // instead of overflow, saturate (but this is not always a good thing)
        total = INT_MAX
    }

    return total;
}
```

# 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 Zero Initialized Pointer

## 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());
```

# Use of Insufficiently Random Values

## Risk

### What might happen

Random values are often used as a mechanism to prevent malicious users from guessing a value, such as a password, encryption key, or session identifier. Depending on what this random value is used for, an attacker would be able to predict the next numbers generated, or previously generated values. This could enable the attacker to hijack another user's session, impersonate another user, or crack an encryption key (depending on what the pseudo-random value was used for).

## Cause

### How does it happen

The application uses a weak method of generating pseudo-random values, such that other numbers could be determined from a relatively small sample size. Since the pseudo-random number generator used is designed for statistically uniform distribution of values, it is approximately deterministic. Thus, after collecting a few generated values (e.g. by creating a few individual sessions, and collecting the sessionids), it would be possible for an attacker to calculate another sessionid.

Specifically, if this pseudo-random value is used in any security context, such as passwords, keys, or secret identifiers, an attacker would be able to predict the next numbers generated, or previously generated values.

## General Recommendations

### How to avoid it

Generic Guidance:

- Whenever unpredicatable numbers are required in a security context, use a cryptographically strong random number generator, instead of a statistical pseudo-random generator.
- Use the cryptorandom generator that is built-in to your language or platform, and ensure it is securely seeded. Do not seed the generator with a weak, non-random seed. (In most cases, the default is securely random).
- Ensure you use a long enough random value, to make brute-force attacks unfeasible.

Specific Recommendations:

- Do not use the statistical pseudo-random number generator, use the cryptorandom generator instead. In Java, this is the SecureRandom class.

## Source Code Examples

### Java

### Use of a weak pseudo-random number generator

```
Random random = new Random();

long sessNum = random.nextLong();
```

```
String sessionId = sessNum.toString();
```

### Cryptographically secure random number generator

```
SecureRandom random = new SecureRandom();

byte sessBytes[] = new byte[32];

random.nextBytes(sessBytes);

String sessionId = new String(sessBytes);
```

## Objc

### Use of a weak pseudo-random number generator

```
long sessNum = rand();
NSString* sessionId = [NSString stringWithFormat:@"%ld", sessNum];
```

### Cryptographically secure random number generator

```
UInt32 sessBytes;
SecRandomCopyBytes(kSecRandomDefault, sizeof(sessBytes), (uint8_t*)&sessBytes);

NSString* sessionId = [NSString stringWithFormat:@"%llu", sessBytes];
```

## Swift

### Use of a weak pseudo-random number generator

```
let sessNum = rand();
let sessionId = String(format:"%ld", sessNum)
```

### Cryptographically secure random number generator

```
var sessBytes: UInt32 = 0
withUnsafeMutablePointer(&sessBytes, { (sessBytesPointer) -> Void in
    let castedPointer = unsafeBitCast(sessBytesPointer, UnsafeMutablePointer<UInt8>.self)
    SecRandomCopyBytes(kSecRandomDefault, sizeof(UInt32), castedPointer)
})

let sessionId = String(format:"%llu", sessBytes)
```

# Unchecked Return Value

## Risk

### What might happen

A program that does not check function return values could cause the application to enter an undefined state. This could lead to unexpected behavior and unintended consequences, including inconsistent data, system crashes or other error-based exploits.

## Cause

### How does it happen

The application calls a system function, but does not receive or check the result of this funciton. These functions often return error codes in the result, or share other status codes with it's caller. The application simply ignores this result value, losing this vital information.

## General Recommendations

### How to avoid it

- Always check the result of any called function that returns a value, and verify the result is an expected value.

- Ensure the calling function responds to all possible return values.

- Expect runtime errors and handle them gracefully. Explicitly define a mechanism for handling unexpected errors.

## Source Code Examples

### CPP

### Unchecked Memory Allocation

```cpp
buff = (char*) malloc(size);
strncpy(buff, source, size);
```

### Safer Memory Allocation

```cpp
buff = (char*) malloc(size+1);
if (buff==NULL) exit(1);

strncpy(buff, source, size);
buff[size] = '\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

# Potential Off by One Error in Loops

## Risk

**What might happen**

An off by one error may result in overwriting or over-reading of unintended memory; in most cases, this can result in unexpected behavior and even application crashes. In other cases, where allocation can be controlled by an attacker, a combination of variable assignment and an off by one error can result in execution of malicious code.

## Cause

**How does it happen**

Often when designating variables to memory, a calculation error may occur when determining size or length that is off by one.

For example in loops, when allocating an array of size 2, its cells are counted as 0,1 - therefore, if a For loop iterator on the array is incorrectly set with the start condition i=0 and the continuation condition i<=2, three cells will be accessed instead of 2, and an attempt will be made to write or read cell [2], which was not originally allocated, resulting in potential corruption of memory outside the bounds of the originally assigned array.

Another example occurs when a null-byte terminated string, in the form of a character array, is copied without its terminating null-byte. Without the null-byte, the string representation is unterminated, resulting in certain functions to over-read memory as they expect the missing null terminator.

## General Recommendations

**How to avoid it**

- Always ensure that a given iteration boundary is correct:
    - With array iterations, consider that arrays begin with cell 0 and end with cell n-1, for a size n array.
    - With character arrays and null-byte terminated string representations, consider that the null byte is required and should not be overwritten or ignored; ensure functions in use are not vulnerable to off-by-one, specifically for instances where null-bytes are automatically appended after the buffer, instead of in place of its last character.
- Where possible, use safe functions that manage memory and are not prone to off-by-one errors.

## Source Code Examples

**CPP**

**Off-By-One in For Loop**

```
int *ptr;
ptr = (int*)malloc(5 * sizeof(int));
for (int i = 0; i <= 5; i++)
{
     ptr[i] = i * 2 + 1; // ptr[5] will be set, but is out of bounds
}
```

### Proper Iteration in For Loop

```
int *ptr;
ptr = (int*)malloc(5 * sizeof(int));
for (int i = 0; i < 5; i++)
{
      ptr[i] = i * 2 + 1; // ptr[0-4] are well defined
}
```

### Off-By-One in strncat

```
strncat(buf, input, sizeof(buf) - strlen(buf)); // actual value should be sizeof(buf)-
strlen(buf)-1 - this form will overwrite the terminating nullbyte
```

# 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

# Client Insufficient ClickJacking Protection

## Risk

### What might happen

Clickjacking attacks allow an attacker to "hijack" a user's mouse clicks on a webpage, by invisibly framing the application, and superimposing it in front of a bogus site. When the user is convinced to click on the bogus website, e.g. on a link or a button, the user's mouse is actually clicking on the target webpage, despite being invisible.

This could allow the attacker to craft an overlay that, when clicked, would lead the user to perform undesirable actions in the vulnerable application, e.g. enabling the user's webcam, deleting all the user's records, changing the user's settings, or causing clickfraud.

## Cause

### How does it happen

The root cause of vulnerability to a clickjacking attack, is that the application's web pages can be loaded into a frame of another website. The application does not implement a proper frame-busting script, that would prevent the page from being loaded into another frame. Note that there are many types of simplistic redirection scripts that still leave the application vulnerable to clickjacking techniques, and should not be used.

When dealing with modern browsers, applications mitigate this vulnerability by issuing appropriate Content-Security-Policy or X-Frame-Options headers to indicate to the browser to disallow framing. However, many legacy browsers do not support this feature, and require a more manual approach by implementing a mitigation in Javascript. To ensure legacy support, a framebusting script is required.

## General Recommendations

### How to avoid it

Generic Guidance:

- Define and implement a a Content Security Policy (CSP) on the server side, including a frame-ancestors directive. Enforce the CSP on all relevant webpages.
- If certain webpages are required to be loaded into a frame, define a specific, whitelisted target URL.
- Alternatively, return a "X-Frame-Options" header on all HTTP responses. If it is necessary to allow a particular webpage to be loaded into a frame, define a specific, whitelisted target URL.
- For legacy support, implement framebusting code using Javascript and CSS to ensure that, if a page is framed, it is never displayed, and attempt to navigate into the frame to prevent attack. Even if navigation fails, the page is not displayed and is therefore not interactive, mitigating potential clickjacking attacks.

Specific Recommendations:

- Implement a proper framebuster script on the client, that is not vulnerable to frame-buster-busting attacks.
  - Code should first disable the UI, such that even if frame-busting is successfully evaded, the UI cannot be clicked. This can be done by setting the CSS value of the "display" attribute to "none" on either the "body" or "html" tags. This is done because, if a frame attempts to redirect and become the parent, the malicious parent can still prevent redirection via various techniques.
  - Code should then determine whether no framing occurs by comparing self === top; if the result is true, can the UI be enabled. If it is false, attempt to navigate away from the framing page by setting the top.location attribute to self.location.

## Source Code Examples

### JavaScript

### Clickjackable Webpage

```
<html>
    <body>
    <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

### Bustable Framebuster

```
<html>
    <head>
    <script>
            if ( window.self.location != window.top.location ) {
                   window.top.location = window.self.location;
            }
        </script>
    </head>

    <body>
        <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

### Proper Framebusterbusterbusting

```
<html>
    <head>
    <style> html {display : none; } </style>
        <script>
            if ( self === top ) {
                   document.documentElement.style.display = 'block';
            }
            else {
                   top.location = self.location;
            }
        </script>
    </head>

    <body>
        <button onclick="clicked();">
            Click here if you love ducks
        </button>
    </body>
</html>
```

## Insecure Temporary File

**Weakness ID:** 377 *(Weakness Base)* **Status:** Incomplete

### Description

#### Description Summary

Creating and using insecure temporary files can leave application and system data vulnerable to attack.

### Time of Introduction

- Architecture and Design
- Implementation

### Applicable Platforms

#### Languages

All

### Demonstrative Examples

#### Example 1

The following code uses a temporary file for storing intermediate data gathered from the network before it is processed.

*(Bad Code)*

*Example Language:* **C**

```
if (tmpnam_r(filename)) {

FILE* tmp = fopen(filename,"wb+");
while((recv(sock,recvbuf,DATA_SIZE, 0) > 0)&(amt!=0)) amt = fwrite(recvbuf,1,DATA_SIZE,tmp);
}
...
```

This otherwise unremarkable code is vulnerable to a number of different attacks because it relies on an insecure method for creating temporary files. The vulnerabilities introduced by this function and others are described in the following sections. The most egregious security problems related to temporary file creation have occurred on Unix-based operating systems, but Windows applications have parallel risks. This section includes a discussion of temporary file creation on both Unix and Windows systems. Methods and behaviors can vary between systems, but the fundamental risks introduced by each are reasonably constant.

### Other Notes

Applications require temporary files so frequently that many different mechanisms exist for creating them in the C Library and Windows(R) API. Most of these functions are vulnerable to various forms of attacks.

The functions designed to aid in the creation of temporary files can be broken into two groups based whether they simply provide a filename or actually open a new file. - Group 1: "Unique" Filenames: The first group of C Library and WinAPI functions designed to help with the process of creating temporary files do so by generating a unique file name for a new temporary file, which the program is then supposed to open. This group includes C Library functions like tmpnam(), tempnam(), mktemp() and their C++ equivalents prefaced with an _ (underscore) as well as the GetTempFileName() function from the Windows API. This group of functions suffers from an underlying race condition on the filename chosen. Although the functions guarantee that the filename is unique at the time it is selected, there is no mechanism to prevent another process or an attacker from creating a file with the same name after it is selected but before the application attempts to open the file. Beyond the risk of a legitimate collision caused by another call to the same function, there is a high probability that an attacker will be able to create a malicious collision because the filenames generated by these functions are not sufficiently randomized to make them difficult to guess. If a file with the selected name is created, then depending on how the file is opened the existing contents or access permissions of the file may remain intact. If the existing contents of the file are malicious in nature, an attacker may be able to inject dangerous data into the application when it reads data back from the temporary file. If an attacker pre-creates the file with relaxed access permissions, then data stored in the temporary file by the application may be accessed, modified or corrupted by an attacker. On Unix based systems an even more insidious attack is possible if the attacker pre-creates the file as a link to another important file. Then, if the application truncates or writes data to the file, it may unwittingly perform damaging operations for the attacker. This is an especially serious threat if the program operates with elevated permissions. Finally, in the best case the file will be opened with the a call to open() using the O_CREAT and O_EXCL flags or to CreateFile() using the CREATE_NEW attribute, which will fail if the file already exists and therefore prevent the types of attacks described above. However, if an attacker is able to accurately predict a sequence of temporary file names, then the application may be prevented from opening necessary temporary storage causing a denial of service (DoS) attack. This type of attack would not be difficult to mount given the small amount of randomness used in the selection of the filenames generated by these functions. - Group 2: "Unique" Files: The second group of C Library functions attempts to resolve some of the security problems related to temporary files by not only generating a unique file name, but also opening the file. This group includes C Library functions like tmpfile() and its C++ equivalents prefaced with an _ (underscore), as well as the slightly better-behaved C Library function mkstemp(). The tmpfile() style functions construct a unique filename and open it in the same way that fopen() would if passed the flags "wb+", that is, as a binary file in read/write mode. If the file already exists, tmpfile() will truncate it to size zero, possibly in an attempt to assuage the security concerns mentioned earlier regarding the race condition that exists between the selection of a supposedly unique filename and the subsequent opening of the selected file. However, this behavior clearly does not solve the function's security problems. First, an attacker can pre-create the file with relaxed access-permissions that will likely be retained by the file opened by tmpfile(). Furthermore, on Unix based systems if the attacker pre-creates the file as a link to another important file, the application may use its possibly elevated permissions to truncate that file, thereby doing damage on behalf of the attacker. Finally, if tmpfile() does create a new file, the access permissions applied to that file will vary from one operating system to another, which can leave application data vulnerable even if an attacker is unable to predict the filename to be used in advance. Finally, mkstemp() is a reasonably safe way create temporary files. It will attempt to create and open a unique file based on a filename template provided by the user combined with a series of randomly generated characters. If it is unable to create such a file, it will fail and return -1. On modern systems the file is opened using mode 0600, which means the file will be secure from tampering unless the user explicitly changes its access permissions. However, mkstemp() still suffers from the use of predictable file names and can leave an application vulnerable to denial of service attacks if an attacker causes mkstemp() to fail by predicting and pre-creating the filenames to be used.

## Relationships

| Nature | Type | ID | Name | View(s) this relationship pertains to |
|---|---|---|---|---|
| ChildOf | Category | 361 | Time and State | **Seven Pernicious Kingdoms (primary)700** |
| ChildOf | Category | 376 | Temporary File Issues | **Development Concepts (primary)699** |
| ChildOf | Weakness Class | 668 | Exposure of Resource to Wrong Sphere | **Research Concepts (primary)1000** |
| ParentOf | Weakness Base | 378 | Creation of Temporary File With Insecure Permissions | **Research Concepts (primary)1000** |
| ParentOf | Weakness Base | 379 | Creation of Temporary File in Directory with Incorrect Permissions | **Research Concepts (primary)1000** |

## Taxonomy Mappings

| Mapped Taxonomy Name | Node ID | Fit | Mapped Node Name |
|---|---|---|---|
| 7 Pernicious Kingdoms | | | Insecure Temporary File |

## References

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 23, "Creating Temporary Files Securely" Page 682. 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 | | |
| 2008-09-08 | CWE Content Team | MITRE | Internal |
| | updated Relationships, Other Notes, Taxonomy Mappings | | |
| 2009-03-10 | CWE Content Team | MITRE | Internal |
| | updated Demonstrative Examples | | |
| 2009-05-27 | CWE Content Team | MITRE | Internal |
| | updated Demonstrative Examples | | |
| 2010-02-16 | CWE Content Team | MITRE | Internal |
| | updated References | | |

BACK TO TOP

## 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

## 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<br>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<br>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<br>Availability<br>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**<br>**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**

```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**<br>**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**<br>Research Concepts1000 |
| ParentOf | Weakness Class | 638 | Failure to Use Complete Mediation | Research Concepts1000 |
| ParentOf | Weakness Base | 804 | Guessable CAPTCHA | **Development Concepts (primary)699**<br>**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

## Incorrect Permission Assignment for Critical Resource

**Weakness ID:** 732 *(Weakness Class)* **Status:** Draft

## Description

### Description Summary

The software specifies permissions for a security-critical resource in a way that allows that resource to be read or modified by unintended actors.

### Extended Description

When a resource is given a permissions setting that provides access to a wider range of actors than required, it could lead to the disclosure of sensitive information, or the modification of that resource by unintended parties. This is especially dangerous when the resource is related to program configuration, execution or sensitive user data.

## Time of Introduction

- Architecture and Design
- Implementation
- Installation
- Operation

## Applicable Platforms

### Languages

Language-independent

## Modes of Introduction

The developer may set loose permissions in order to minimize problems when the user first runs the program, then create documentation stating that permissions should be tightened. Since system administrators and users do not always read the documentation, this can result in insecure permissions being left unchanged.

The developer might make certain assumptions about the environment in which the software runs - e.g., that the software is running on a single-user system, or the software is only accessible to trusted administrators. When the software is running in a different environment, the permissions become a problem.

## Common Consequences

| Scope | Effect |
|---|---|
| Confidentiality | An attacker may be able to read sensitive information from the associated resource, such as credentials or configuration information stored in a file. |
| Integrity | An attacker may be able to modify critical properties of the associated resource to gain privileges, such as replacing a world-writable executable with a Trojan horse. |
| Availability | An attacker may be able to destroy or corrupt critical data in the associated resource, such as deletion of records from a database. |

## Likelihood of Exploit

Medium to High

## Detection Methods

### Automated Static Analysis

Automated static analysis may be effective in detecting permission problems for system resources such as files, directories, shared memory, device interfaces, etc. Automated techniques may be able to detect the use of library functions that modify permissions, then analyze function calls for arguments that contain potentially insecure values.

However, since the software's intended security policy might allow loose permissions for certain operations (such as publishing a file on a web server), automated static analysis may produce some false positives - i.e., warnings that do not have any security consequences or require any code changes.

When custom permissions models are used - such as defining who can read messages in a particular forum in a bulletin board system - these can be difficult to detect using automated static analysis. It may be possible to define custom signatures that identify any custom functions that implement the permission checks and assignments.

### Automated Dynamic Analysis

Automated dynamic analysis may be effective in detecting permission problems for system resources such as files, directories, shared memory, device interfaces, etc.

However, since the software's intended security policy might allow loose permissions for certain operations (such as publishing a file on a web server), automated dynamic analysis may produce some false positives - i.e., warnings that do not have any security consequences or require any code changes.

When custom permissions models are used - such as defining who can read messages in a particular forum in a bulletin board system - these can be difficult to detect using automated dynamic analysis. It may be possible to define custom signatures that identify any custom functions that implement the permission checks and assignments.

### Manual Static Analysis

Manual static analysis may be effective in detecting the use of custom permissions models and functions. The code could then be examined to identifying usage of the related functions. Then the human analyst could evaluate permission assignments in the context of the intended security model of the software.

### Manual Dynamic Analysis

Manual dynamic analysis may be effective in detecting the use of custom permissions models and functions. The program could then be executed with a focus on exercising code paths that are related to the custom permissions. Then the human analyst could evaluate permission assignments in the context of the intended security model of the software.

### Fuzzing

Fuzzing is not effective in detecting this weakness.

## Demonstrative Examples

## Example 1

The following code sets the umask of the process to 0 before creating a file and writing "Hello world" into the file.

*(Bad Code)*

*Example Language:* **C**

```
#define OUTFILE "hello.out"

umask(0);
FILE *out;
/* Ignore CWE-59 (link following) for brevity */
out = fopen(OUTFILE, "w");
if (out) {
fprintf(out, "hello world!\n");
fclose(out);
}
```

After running this program on a UNIX system, running the "ls -l" command might return the following output:

*(Result)*

```
-rw-rw-rw- 1 username 13 Nov 24 17:58 hello.out
```

The "rw-rw-rw-" string indicates that the owner, group, and world (all users) can read the file and write to it.

## Example 2

The following code snippet might be used as a monitor to periodically record whether a web site is alive. To ensure that the file can always be modified, the code uses chmod() to make the file world-writable.

*(Bad Code)*

*Example Language:* **Perl**

```
$fileName = "secretFile.out";

if (-e $fileName) {
chmod 0777, $fileName;
}
```

```
my $outFH;
if (! open($outFH, ">>$fileName")) {
ExitError("Couldn't append to $fileName: $!");
}
my $dateString = FormatCurrentTime();
my $status = IsHostAlive("cwe.mitre.org");
print $outFH "$dateString cwe status: $status!\n";
close($outFH);
```

The first time the program runs, it might create a new file that inherits the permissions from its environment. A file listing might look like:

*(Result)*

```
-rw-r--r-- 1 username 13 Nov 24 17:58 secretFile.out
```

This listing might occur when the user has a default umask of 022, which is a common setting. Depending on the nature of the file, the user might not have intended to make it readable by everyone on the system.

The next time the program runs, however - and all subsequent executions - the chmod will set the file's permissions so that the owner, group, and world (all users) can read the file and write to it:

*(Result)*

```
-rw-rw-rw- 1 username 13 Nov 24 17:58 secretFile.out
```

Perhaps the programmer tried to do this because a different process uses different permissions that might prevent the file from being updated.

## Example 3

The following command recursively sets world-readable permissions for a directory and all of its children:

*(Bad Code)*

*Example Language:* **Shell**

```
chmod -R ugo+r DIRNAME
```

If this command is run from a program, the person calling the program might not expect that all the files under the directory will be world-readable. If the directory is expected to contain private data, this could become a security problem.

### Observed Examples

| Reference | Description |
|---|---|
| CVE-2009-3482 | Anti-virus product sets insecure "Everyone: Full Control" permissions for files under the "Program Files" folder, allowing attackers to replace executables with Trojan horses. |
| CVE-2009-3897 | Product creates directories with 0777 permissions at installation, allowing users to gain privileges and access a socket used for authentication. |
| CVE-2009-3489 | Photo editor installs a service with an insecure security descriptor, allowing users to stop or start the service, or execute commands as SYSTEM. |
| CVE-2009-3289 | Library function copies a file to a new target and uses the source file's permissions for the target, which is incorrect when the source file is a symbolic link, which typically has 0777 permissions. |
| CVE-2009-0115 | Device driver uses world-writable permissions for a socket file, allowing attackers to inject arbitrary commands. |
| CVE-2009-1073 | LDAP server stores a cleartext password in a world-readable file. |
| CVE-2009-0141 | Terminal emulator creates TTY devices with world-writable permissions, allowing an attacker to write to the terminals of other users. |

| | |
|---|---|
| CVE-2008-0662 | VPN product stores user credentials in a registry key with "Everyone: Full Control" permissions, allowing attackers to steal the credentials. |
| CVE-2008-0322 | Driver installs its device interface with "Everyone: Write" permissions. |
| CVE-2009-3939 | Driver installs a file with world-writable permissions. |
| CVE-2009-3611 | Product changes permissions to 0777 before deleting a backup; the permissions stay insecure for subsequent backups. |
| CVE-2007-6033 | Product creates a share with "Everyone: Full Control" permissions, allowing arbitrary program execution. |
| CVE-2007-5544 | Product uses "Everyone: Full Control" permissions for memory-mapped files (shared memory) in inter-process communication, allowing attackers to tamper with a session. |
| CVE-2005-4868 | Database product uses read/write permissions for everyone for its shared memory, allowing theft of credentials. |
| CVE-2004-1714 | Security product uses "Everyone: Full Control" permissions for its configuration files. |
| CVE-2001-0006 | "Everyone: Full Control" permissions assigned to a mutex allows users to disable network connectivity. |
| CVE-2002-0969 | Chain: database product contains buffer overflow that is only reachable through a .ini configuration file - which has "Everyone: Full Control" permissions. |

## Potential Mitigations

### Phase: Implementation

When using a critical resource such as a configuration file, check to see if the resource has insecure permissions (such as being modifiable by any regular user), and generate an error or even exit the software if there is a possibility that the resource could have been modified by an unauthorized party.

### Phase: Architecture and Design

Divide your application into anonymous, normal, privileged, and administrative areas. Reduce the attack surface by carefully defining distinct user groups, privileges, and/or roles. Map these against data, functionality, and the related resources. Then set the permissions accordingly. This will allow you to maintain more fine-grained control over your resources.

### Phases: Implementation; Installation

During program startup, explicitly set the default permissions or umask to the most restrictive setting possible. Also set the appropriate permissions during program installation. This will prevent you from inheriting insecure permissions from any user who installs or runs the program.

### Phase: System Configuration

For all configuration files, executables, and libraries, make sure that they are only readable and writable by the software's administrator.

### Phase: Documentation

Do not suggest insecure configuration changes in your documentation, especially if those configurations can extend to resources and other software that are outside the scope of your own software.

### Phase: Installation

Do not assume that the system administrator will manually change the configuration to the settings that you recommend in the manual.

### Phase: Testing

Use 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. These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules.

### Phase: Testing

Use monitoring tools that examine the software's process as it interacts with the operating system and the network. This technique is useful in cases when source code is unavailable, if the software was not developed by you, or if you want to verify that the build phase did not introduce any new weaknesses. Examples include debuggers that directly attach to the running process; system-call tracing utilities such as truss (Solaris) and strace (Linux); system activity monitors such as FileMon, RegMon, Process Monitor, and other Sysinternals utilities (Windows); and sniffers and protocol analyzers that monitor network traffic.

Attach the monitor to the process and watch for library functions or system calls on OS resources such as files, directories, and shared memory. Examine the arguments to these calls to infer which permissions are being used.

Note that this technique is only useful for permissions issues related to system resources. It is not likely to detect application-level business rules that are related to permissions, such as if a user of a blog system marks a post as "private," but the blog system inadvertently marks it as "public."

**Phases: Testing; System Configuration**

Ensure that your software runs properly under the Federal Desktop Core Configuration (FDCC) or an equivalent hardening configuration guide, which many organizations use to limit the attack surface and potential risk of deployed software.

## Relationships

| Nature | Type | ID | Name | View(s) this relationship pertains to |
|---|---|---|---|---|
| ChildOf | Category | 275 | Permission Issues | **Development Concepts (primary)699** |
| ChildOf | Weakness Class | 668 | Exposure of Resource to Wrong Sphere | **Research Concepts (primary)1000** |
| 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** |
| RequiredBy | Compound Element: Composite | 689 | Permission Race Condition During Resource Copy | Research Concepts1000 |
| ParentOf | Weakness Variant | 276 | Incorrect Default Permissions | **Research Concepts (primary)1000** |
| ParentOf | Weakness Variant | 277 | Insecure Inherited Permissions | **Research Concepts (primary)1000** |
| ParentOf | Weakness Variant | 278 | Insecure Preserved Inherited Permissions | **Research Concepts (primary)1000** |
| ParentOf | Weakness Variant | 279 | Incorrect Execution-Assigned Permissions | **Research Concepts (primary)1000** |
| ParentOf | Weakness Base | 281 | Improper Preservation of Permissions | **Research Concepts (primary)1000** |

## Related Attack Patterns

| CAPEC-ID | Attack Pattern Name | *(CAPEC Version: 1.5)* |
|---|---|---|
| 232 | Exploitation of Privilege/Trust | |
| 1 | Accessing Functionality Not Properly Constrained by ACLs | |
| 17 | Accessing, Modifying or Executing Executable Files | |
| 60 | Reusing Session IDs (aka Session Replay) | |
| 61 | Session Fixation | |
| 62 | Cross Site Request Forgery (aka Session Riding) | |
| 122 | Exploitation of Authorization | |
| 180 | Exploiting Incorrectly Configured Access Control Security Levels | |
| 234 | Hijacking a privileged process | |

## References

Mark Dowd, John McDonald and Justin Schuh. "The Art of Software Security Assessment". Chapter 9, "File Permissions." Page 495.. 1st Edition. Addison Wesley. 2006.

John Viega and Gary McGraw. "Building Secure Software". Chapter 8, "Access Control." Page 194.. 1st Edition. Addison-Wesley. 2002.

### Maintenance Notes

The relationships between privileges, permissions, and actors (e.g. users and groups) need further refinement within the Research view. One complication is that these concepts apply to two different pillars, related to control of resources (CWE-664) and protection mechanism failures (CWE-396).

## Content History

### Submissions

| Submission Date | Submitter | Organization | Source |
|---|---|---|---|
| 2008-09-08 | | | Internal CWE Team |
| | new weakness-focused entry for Research view. | | |

### Modifications

| Modification Date | Modifier | Organization | Source |
|---|---|---|---|
| 2009-01-12 | CWE Content Team | MITRE | Internal |
| | updated Description, Likelihood of Exploit, Name, Potential Mitigations, Relationships | | |
| 2009-03-10 | CWE Content Team | MITRE | Internal |
| | updated Potential Mitigations, Related Attack Patterns | | |
| 2009-05-27 | CWE Content Team | MITRE | Internal |
| | updated Name | | |
| 2009-12-28 | CWE Content Team | MITRE | Internal |
| | updated Applicable Platforms, Common Consequences, Demonstrative Examples, Detection Factors, Modes of Introduction, Observed Examples, Potential Mitigations, References | | |
| 2010-02-16 | CWE Content Team | MITRE | Internal |
| | updated Relationships | | |
| 2010-04-05 | CWE Content Team | MITRE | Internal |
| | updated Potential Mitigations, Related Attack Patterns | | |

### Previous Entry Names

| Change Date | Previous Entry Name |
|---|---|
| 2009-01-12 | Insecure Permission Assignment for Resource |
| 2009-05-27 | Insecure Permission Assignment for Critical Resource |

BACK TO TOP

# Exposure of System Data to Unauthorized Control Sphere

## Risk

### What might happen

System data can provide attackers with valuable insights on systems and services they are targeting - any type of system data, from service version to operating system fingerprints, can assist attackers to hone their attack, correlate data with known vulnerabilities or focus efforts on developing new attacks against specific technologies.

## Cause

### How does it happen

System data is read and subsequently exposed where it might be read by untrusted entities.

## General Recommendations

### How to avoid it

Consider the implications of exposure of the specified input, and expected level of access to the specified output. If not required, consider removing this code, or modifying exposed information to exclude potentially sensitive system data.

## Source Code Examples

### Java

### Leaking Environment Variables in JSP Web-Page

```
String envVarValue = System.getenv(envVar);
if (envVarValue == null) {
    out.println("Environment variable is not defined:");
    out.println(System.getenv());
} else {
    //[..]
};
```

# TOCTOU

## Risk

### What might happen

At best, a Race Condition may cause errors in accuracy, overidden values or unexpected behavior that may result in denial-of-service. At worst, it may allow attackers to retrieve data or bypass security processes by replaying a controllable Race Condition until it plays out in their favor.

## Cause

### How does it happen

Race Conditions occur when a public, single instance of a resource is used by multiple concurrent logical processes. If the these logical processes attempt to retrieve and update the resource without a timely management system, such as a lock, a Race Condition will occur.

An example for when a Race Condition occurs is a resource that may return a certain value to a process for further editing, and then updated by a second process, resulting in the original process' data no longer being valid. Once the original process edits and updates the incorrect value back into the resource, the second process' update has been overwritten and lost.

## General Recommendations

### How to avoid it

When sharing resources between concurrent processes across the application ensure that these resources are either thread-safe, or implement a locking mechanism to ensure expected concurrent activity.

## Source Code Examples

### Java

### Different Threads Increment and Decrement The Same Counter Repeatedly, Resulting in a Race Condition

```
        public static int counter = 0;
        public static void start() throws InterruptedException {
                incrementCounter ic;
                decrementCounter dc;
                while(counter == 0) {
                        counter = 0;
                        ic = new incrementCounter();
                        dc = new decrementCounter();
                        ic.start();
                        dc.start();
                        ic.join();
                        dc.join();
                }
                System.out.println(counter); //Will stop and return either -1 or 1 due to race
  condition over counter
        }

        public static class incrementCounter extends Thread {
            public void run() {
                counter++;
            }
```

```
}

public static class decrementCounter extends Thread {
    public void run() {
       counter--;
    }
}
```

### Different Threads Increment and Decrement The Same Thread-Safe Counter Repeatedly, Never Resulting in a Race Condition

```
public static int counter = 0;
public static Object lock = new Object();

public static void start() throws InterruptedException {
       incrementCounter ic;
       decrementCounter dc;
       while(counter == 0) { // because of proper locking, this condition is never false
              counter = 0;
              ic = new incrementCounter();
              dc = new decrementCounter();
              ic.start();
              dc.start();
              ic.join();
              dc.join();
       }
       System.out.println(counter); // Never reached
}

public static class incrementCounter extends Thread {
    public void run() {
       synchronized (lock) {
              counter++;
       }
    }
}

public static class decrementCounter extends Thread {
    public void run() {
       synchronized (lock) {
              counter--;
       }
    }
}
```

## Scanned Languages

| Language | Hash Number | Change Date |
|---|---|---|
| CPP | 4541647240435660 | 6/19/2024 |
| JavaScript | 0831671993165974 | 6/19/2024 |
| VbScript | 1349101913133594 | 6/19/2024 |
| Typescript | 2047548555014888 | 6/19/2024 |
| Common | 0105849645654507 | 6/19/2024 |