

## EasyLogger Scan Report

Project Name	EasyLogger
Scan Start	Friday, June 21, 2024 11:26:44 PM
Preset	Checkmarx Default
Scan Time	00h:10m:11s
Lines Of Code Scanned	1814
Files Scanned	3
Report Creation Time	Friday, June 21, 2024 11:41:42 PM
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067</a>
Team	CxServer
Checkmarx Version	8.7.0
Scan Type	Full
Source Origin	LocalPath
Density	3/100 (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**

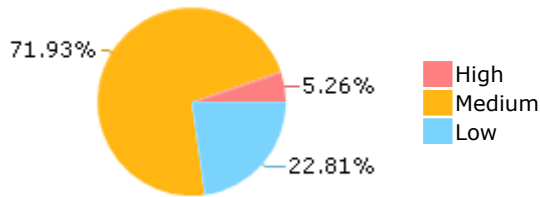
Results limit per query was set to 50

**Selected Queries**

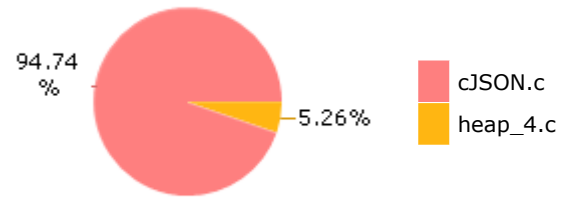
Selected queries are listed in [Result Summary](#)

---

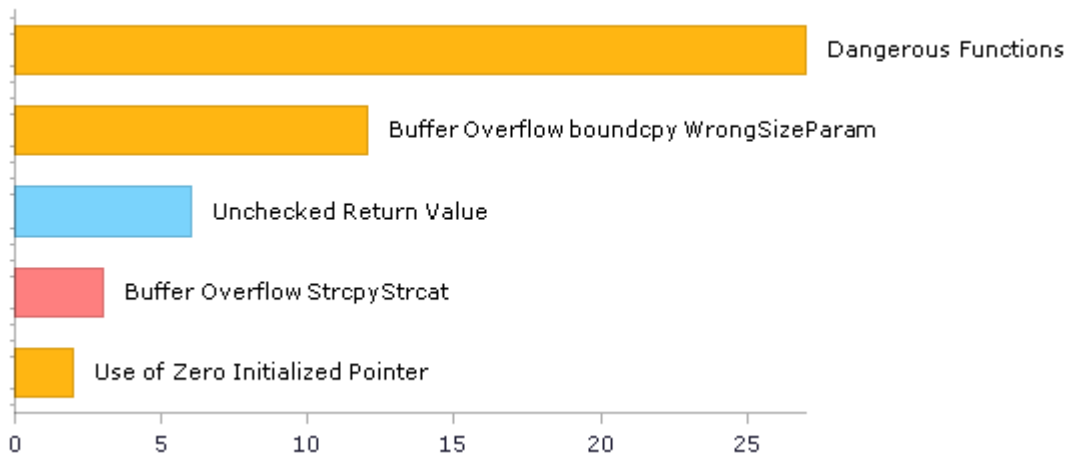
## Result Summary



## 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	16	14
A2-Broken Authentication	App. Specific	EASY	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A3-Sensitive Data Exposure	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App. Specific	0	0
A4-XML External Entities (XXE)	App. Specific	AVERAGE	COMMON	EASY	SEVERE	App. Specific	0	0
A5-Broken Access Control*	App. Specific	AVERAGE	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A6-Security Misconfiguration	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A7-Cross-Site Scripting (XSS)	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A8-Insecure Deserialization	App. Specific	DIFFICULT	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A9-Using Components with Known Vulnerabilities*	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	MODERATE	App. Specific	27	27
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	27	27
A10-Unvalidated Redirects and Forwards	USERS BROWSERS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0

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

## Scan Summary - PCI DSS v3.2

Category	Issues Found	Best Fix Locations
PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection	0	0
PCI DSS (3.2) - 6.5.2 - Buffer overflows	15	13
PCI DSS (3.2) - 6.5.3 - Insecure cryptographic storage	0	0
PCI DSS (3.2) - 6.5.4 - Insecure communications	0	0
PCI DSS (3.2) - 6.5.5 - Improper error handling*	0	0
PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)	0	0
PCI DSS (3.2) - 6.5.8 - Improper access control	0	0
PCI DSS (3.2) - 6.5.9 - Cross-site request forgery	0	0
PCI DSS (3.2) - 6.5.10 - Broken authentication and session management	0	0

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

## Scan Summary - FISMA 2014

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

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

## Scan Summary - NIST SP 800-53

Category	Issues Found	Best Fix Locations
AC-12 Session Termination (P2)	0	0
AC-3 Access Enforcement (P1)	0	0
AC-4 Information Flow Enforcement (P1)	0	0
AC-6 Least Privilege (P1)	0	0
AU-9 Protection of Audit Information (P1)	0	0
CM-6 Configuration Settings (P2)	0	0
IA-5 Authenticator Management (P1)	0	0
IA-6 Authenticator Feedback (P2)	0	0
IA-8 Identification and Authentication (Non-Organizational Users) (P1)	0	0
SC-12 Cryptographic Key Establishment and Management (P1)	0	0
SC-13 Cryptographic Protection (P1)	0	0
SC-17 Public Key Infrastructure Certificates (P1)	0	0
SC-18 Mobile Code (P2)	0	0
SC-23 Session Authenticity (P1)*	0	0
SC-28 Protection of Information at Rest (P1)	0	0
SC-4 Information in Shared Resources (P1)	0	0
SC-5 Denial of Service Protection (P1)*	3	3
SC-8 Transmission Confidentiality and Integrity (P1)	0	0
SI-10 Information Input Validation (P1)*	3	1
SI-11 Error Handling (P2)*	6	6
SI-15 Information Output Filtering (P0)	0	0
SI-16 Memory Protection (P1)	0	0

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



## Scan Summary - OWASP Mobile Top 10 2016

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

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

## Scan Summary - Custom

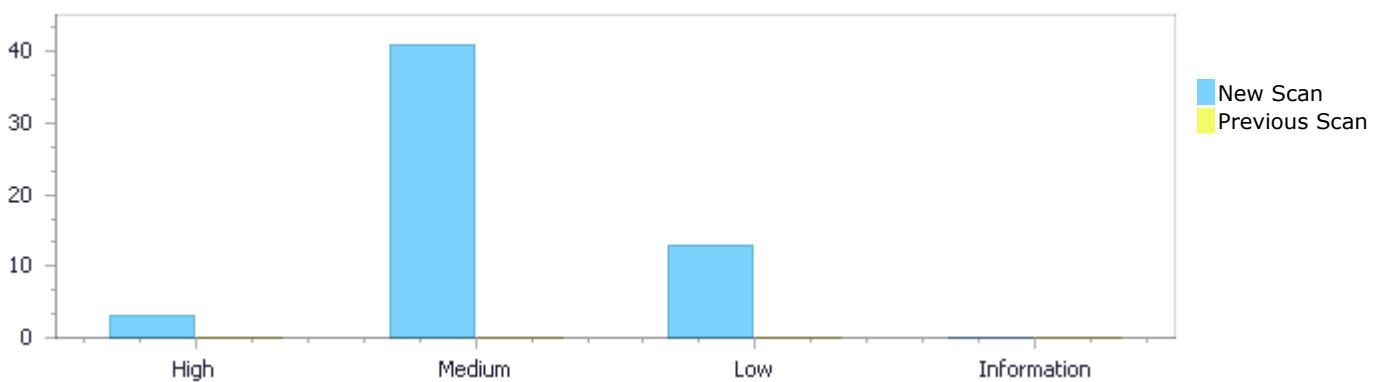
Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

## Results Distribution By Status

First scan of the project

	High	Medium	Low	Information	Total
New Issues	3	41	13	0	57
Recurrent Issues	0	0	0	0	0
Total	3	41	13	0	57

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	3	41	13	0	57
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	3	41	13	0	57

## Result Summary

Vulnerability Type	Occurrences	Severity
<a href="#">Buffer Overflow StrcpyStrcat</a>	3	High
<a href="#">Dangerous Functions</a>	27	Medium
<a href="#">Buffer Overflow boundcpy WrongSizeParam</a>	12	Medium
<a href="#">Use of Zero Initialized Pointer</a>	2	Medium
<a href="#">Unchecked Return Value</a>	6	Low

<a href="#">Use of Sizeof On a Pointer Type</a>	6	Low
<a href="#">NULL Pointer Dereference</a>	1	Low

## 10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
EasyLogger/cJSON.c	42
EasyLogger/heap_4.c	2

# Scan Results Details

## Buffer Overflow StrcpyStrcat

Query Path:

CPP\Cx\CPP Buffer Overflow\Buffer Overflow StrcpyStrcat Version:1

### Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows  
NIST SP 800-53: SI-10 Information Input Validation (P1)  
OWASP Top 10 2017: A1-Injection

### Description

#### Buffer Overflow StrcpyStrcat\Path 1:

Severity	High
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=1">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=1</a>
Status	New

The size of the buffer used by \*print\_string\_ptr in str, at line 253 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_string\_ptr passes to str, at line 253 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	253	265
Object	str	str

### Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_string\_ptr(const char \*str, printbuffer \*p)

```
....
253. static char *print_string_ptr(const char *str, printbuffer *p)
....
265. strcpy(ptr2, str);
```

#### Buffer Overflow StrcpyStrcat\Path 2:

Severity	High
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=2">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=2</a>
Status	New

The size of the buffer used by \*print\_string\_ptr in ptr2, at line 253 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_string\_ptr passes to str, at line 253 of EasyLogger/cJSON.c, to overwrite the target buffer.

Source	Destination
--------	-------------

File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	253	265
Object	str	ptr2

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_string\_ptr(const char \*str, printbuffer \*p)

```
....
253. static char *print_string_ptr(const char *str, printbuffer *p)
....
265. strcpy(ptr2, str);
```

### Buffer Overflow StrcpyStrcat\Path 3:

Severity High

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=3>

Status New

The size of the buffer used by \*print\_string\_ptr in out, at line 253 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_string\_ptr passes to str, at line 253 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	253	276
Object	str	out

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_string\_ptr(const char \*str, printbuffer \*p)

```
....
253. static char *print_string_ptr(const char *str, printbuffer *p)
....
276. strcpy(out, "\\\"");
```

## 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	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=29">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=29</a>
Status	New

The dangerous function, memcpy, was found in use at line 49 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	56	56
Object	memcpy	memcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char\* cJSON\_strdup(const char\* str)

```
....  
56.         memcpy(copy, str, len);
```

### Dangerous Functions\Path 2:

Severity	Medium
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=30">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=30</a>
Status	New

The dangerous function, memcpy, was found in use at line 121 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	131	131
Object	memcpy	memcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char\* ensure(printbuffer \*p,int needed)

```
....  
131.         if (newbuffer) memcpy(newbuffer, p->buffer, p->length);
```

### Dangerous Functions\Path 3:

Severity	Medium
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=31">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=31</a>



Status New

The dangerous function, memcpy, was found in use at line 435 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	504	504
Object	memcpy	memcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_array(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
504.      tmplen=strlen(entries[i]);memcpy(ptr,entries[i],tmplen);ptr+=tmplen;  
n;
```

#### Dangerous Functions\Path 4:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=32>

Status New

The dangerous function, memcpy, was found in use at line 551 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	646	646
Object	memcpy	memcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
646.      tmplen=strlen(names[i]);memcpy(ptr,names[i],tmplen);ptr+=tmplen;
```

#### Dangerous Functions\Path 5:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=33>

Status	New
--------	-----

The dangerous function, memcpy, was found in use at line 680 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	680	680
Object	memcpy	memcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static cJSON \*create\_reference(cJSON \*item) {cJSON \*ref=cJSON\_New\_Item();if (!ref) return 0;memcpy(ref,item,sizeof(cJSON));ref->string=0;ref->type|=cJSON\_IsReference;ref->next=ref->prev=0;return ref;}

```
....
680. static cJSON *create_reference(cJSON *item) {cJSON
*ref=cJSON_New_Item();if (!ref) return
0;memcpy(ref,item,sizeof(cJSON));ref->string=0;ref-
>type|=cJSON_IsReference;ref->next=ref->prev=0;return ref;}
```

#### Dangerous Functions\Path 6:

Severity	Medium
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=34">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=34</a>
Status	New

The dangerous function, sprintf, was found in use at line 147 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	161	161
Object	sprintf	sprintf

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_number(cJSON \*item,printbuffer \*p)

```
....
161. if (str) sprintf(str,"%d",item->valueint);
```

#### Dangerous Functions\Path 7:

Severity	Medium
Result State	To Verify
Online Results	<a href="http://WIN-">http://WIN-</a>

[BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=35](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=35)

Status New

The dangerous function, printf, was found in use at line 147 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	169	169
Object	printf	printf

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_number(cJSON \*item, printbuffer \*p)

```
....  
169.                if (fpclassify(d) != FP_ZERO && !isnormal(d))  
                    printf(str, "null");
```

#### Dangerous Functions\Path 8:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=36>

Status New

The dangerous function, printf, was found in use at line 147 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	170	170
Object	printf	printf

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_number(cJSON \*item, printbuffer \*p)

```
....  
170.                else if (fabs(floor(d)-d)<=DBL_EPSILON &&  
fabs(d)<1.0e60)    printf(str, "%.0f", d);
```

#### Dangerous Functions\Path 9:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=36>

[67&pathid=37](#)

Status New

The dangerous function, sprintf, was found in use at line 147 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	171	171
Object	sprintf	sprintf

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_number(cJSON \*item, printbuffer \*p)

```
....  
171.                                else if (fabs(d)<1.0e-6 || fabs(d)>1.0e9)  
                                sprintf(str,"%e",d);
```

#### Dangerous Functions\Path 10:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=38>

Status New

The dangerous function, sprintf, was found in use at line 147 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	172	172
Object	sprintf	sprintf

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_number(cJSON \*item, printbuffer \*p)

```
....  
172.                                else  
                                sprintf(str,"%f",d);
```

#### Dangerous Functions\Path 11:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=39>

Status New

The dangerous function, `sprintf`, was found in use at line 253 in `EasyLogger/cJSON.c` file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	302	302
Object	sprintf	sprintf

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_string\_ptr(const char \*str, printbuffer \*p)

```
....  
302.                                     default:  
sprintf(ptr2, "u%04x", token); ptr2+=5;      break;      /* escape and  
print */
```

#### Dangerous Functions\Path 12:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=40>

Status New

The dangerous function, `strcpy`, was found in use at line 147 in `EasyLogger/cJSON.c` file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	155	155
Object	strcpy	strcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_number(cJSON \*item, printbuffer \*p)

```
....  
155.                                     if (str) strcpy(str, "0");
```

#### Dangerous Functions\Path 13:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=41>

Status New

The dangerous function, strcpy, was found in use at line 253 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	265	265
Object	strcpy	strcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_string\_ptr(const char \*str, printbuffer \*p)

```
....  
265.                strcpy(ptr2, str);
```

#### Dangerous Functions\Path 14:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=42>

Status New

The dangerous function, strcpy, was found in use at line 253 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	276	276
Object	strcpy	strcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_string\_ptr(const char \*str, printbuffer \*p)

```
....  
276.                strcpy(out, "\\\"");
```

#### Dangerous Functions\Path 15:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=43>

Status New

The dangerous function, strcpy, was found in use at line 373 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	381	381
Object	strcpy	strcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_value(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
.....
381.                                     case cJSON_NULL: {out=ensure(p,5); if (out)
strcpy(out,"null");                     break;}
```

### Dangerous Functions\Path 16:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=44>

Status New

The dangerous function, strcpy, was found in use at line 373 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	382	382
Object	strcpy	strcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_value(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
.....
382.                                     case cJSON_False: {out=ensure(p,6); if (out)
strcpy(out,"false");                     break;}
```

### Dangerous Functions\Path 17:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=45>

Status New

The dangerous function, strcpy, was found in use at line 373 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

Source	Destination
--------	-------------

File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	383	383
Object	strcpy	strcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_value(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
383.                                case cJSON_True:  {out=ensure(p,5); if (out)  
strcpy(out,"true");              break;}
```

#### Dangerous Functions\Path 18:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=46>

Status New

The dangerous function, strcpy, was found in use at line 435 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	450	450
Object	strcpy	strcpy

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_array(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
450.                                if (out) strcpy(out,"[]");
```

#### Dangerous Functions\Path 19:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=47>

Status New

The dangerous function, strcpy, was found in use at line 551 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c



Line	648	648
Object	strcpy	strcpy

**Code Snippet**

File Name EasyLogger/cJSON.c

Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
648.                strcpy(ptr,entries[i]);ptr+=strlen(entries[i]);
```

**Dangerous Functions\Path 20:**

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=48>

Status New

The dangerous function, strlen, was found in use at line 49 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	54	54
Object	strlen	strlen

**Code Snippet**

File Name EasyLogger/cJSON.c

Method static char\* cJSON\_strdup(const char\* str)

```
....  
54.        len = strlen(str) + 1;
```

**Dangerous Functions\Path 21:**

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=49>

Status New

The dangerous function, strlen, was found in use at line 138 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	143	143
Object	strlen	strlen

## Code Snippet

File Name EasyLogger/cJSON.c  
Method static int update(printbuffer \*p)

```
....  
143.         return p->offset+strlen(str);
```

**Dangerous Functions\Path 22:**

Severity Medium  
Result State To Verify  
Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=50>  
Status New

The dangerous function, strlen, was found in use at line 435 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	482	482
Object	strlen	strlen

## Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_array(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
482.         if (ret) len+=strlen(ret)+2+(fmt?1:0); else  
fail=1;
```

**Dangerous Functions\Path 23:**

Severity Medium  
Result State To Verify  
Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=51>  
Status New

The dangerous function, strlen, was found in use at line 435 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	504	504
Object	strlen	strlen

## Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_array(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
504.          tmplen=strlen(entries[i]);memcpy(ptr,entries[i],tmplen);ptr+=tmple  
n;
```

#### Dangerous Functions\Path 24:

Severity Medium  
Result State To Verify  
Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=52>  
Status New

The dangerous function, strlen, was found in use at line 551 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	625	625
Object	strlen	strlen

#### Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
625.          if (str && ret)  
len+=strlen(ret)+strlen(str)+2+(fmt?2+depth:0); else fail=1;
```

#### Dangerous Functions\Path 25:

Severity Medium  
Result State To Verify  
Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=53>  
Status New

The dangerous function, strlen, was found in use at line 551 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	625	625
Object	strlen	strlen

#### Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
625.                                     if (str && ret)  
len+=strlen(ret)+strlen(str)+2+(fmt?2+depth:0); else fail=1;
```

#### Dangerous Functions\Path 26:

Severity Medium  
Result State To Verify  
Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=54>  
Status New

The dangerous function, strlen, was found in use at line 551 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	646	646
Object	strlen	strlen

#### Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
646.                                     tmplen=strlen(names[i]);memcpy(ptr,names[i],tmplen);ptr+=tmplen;
```

#### Dangerous Functions\Path 27:

Severity Medium  
Result State To Verify  
Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=55>  
Status New

The dangerous function, strlen, was found in use at line 551 in EasyLogger/cJSON.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	648	648
Object	strlen	strlen

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....
648. strcpy(ptr,entries[i]);ptr+=strlen(entries[i]);
```

## 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	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=16">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=16</a>
Status	New

The size of the buffer used by \*create\_reference in cJSON, at line 680 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*create\_reference passes to cJSON, at line 680 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	680	680
Object	cJSON	cJSON

### Code Snippet

File Name EasyLogger/cJSON.c

Method static cJSON \*create\_reference(cJSON \*item) {cJSON \*ref=cJSON\_New\_Item();if (!ref) return 0;memcpy(ref,item,sizeof(cJSON));ref->string=0;ref->type|=cJSON\_IsReference;ref->next=ref->prev=0;return ref;}

```
....
680. static cJSON *create_reference(cJSON *item) {cJSON
*ref=cJSON_New_Item();if (!ref) return
0;memcpy(ref,item,sizeof(cJSON));ref->string=0;ref-
>type|=cJSON_IsReference;ref->next=ref->prev=0;return ref;}
```

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

Severity	Medium
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=17">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=17</a>
Status	New

The size of the buffer used by \*cJSON\_New\_Item in cJSON, at line 73 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source

buffer that \*cJSON\_New\_Item passes to cJSON, at line 73 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	76	76
Object	cJSON	cJSON

#### Code Snippet

File Name EasyLogger/cJSON.c  
Method static cJSON \*cJSON\_New\_Item(void)

```
....
76.    if (node) memset (node,0,sizeof (cJSON));
```

### Buffer Overflow boundcpy WrongSizeParam\Path 3:

Severity	Medium
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=18">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=18</a>
Status	New

The size of the buffer used by \*print\_array in numentries, at line 435 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_array passes to numentries, at line 435 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	475	475
Object	numentries	numentries

#### Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_array(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....
475.    memset (entries,0,numentries*sizeof (char));
```

### Buffer Overflow boundcpy WrongSizeParam\Path 4:

Severity	Medium
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=19">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=19</a>
Status	New

The size of the buffer used by \*print\_array in char, at line 435 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_array passes to char, at line 435 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	475	475
Object	char	char

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_array(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....
475.                memset(entries,0,numentries*sizeof(char*));
```

### Buffer Overflow boundcpy WrongSizeParam\Path 5:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=20>

Status New

The size of the buffer used by \*print\_object in numentries, at line 551 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_object passes to numentries, at line 551 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	616	616
Object	numentries	numentries

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....
616.                memset(entries,0,sizeof(char*)*numentries);
```

### Buffer Overflow boundcpy WrongSizeParam\Path 6:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=21>

Status New

The size of the buffer used by \*print\_object in char, at line 551 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_object passes to char, at line 551 of EasyLogger/cJSON.c, to overwrite the target buffer.

Source	Destination
--------	-------------

File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	616	616
Object	char	char

#### Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....
616.          memset (entries,0,sizeof (char*) *numentries);
```

### Buffer Overflow boundcpy WrongSizeParam\Path 7:

Severity	Medium
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=22">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=22</a>
Status	New

The size of the buffer used by \*print\_object in numentries, at line 551 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_object passes to numentries, at line 551 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	617	617
Object	numentries	numentries

#### Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....
617.          memset (names,0,sizeof (char*) *numentries);
```

### Buffer Overflow boundcpy WrongSizeParam\Path 8:

Severity	Medium
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=23">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=23</a>
Status	New

The size of the buffer used by \*print\_object in char, at line 551 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_object passes to char, at line 551 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c



Line	617	617
Object	char	char

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....
617.             memset(names,0,sizeof(char*) *numentries);
```

### Buffer Overflow boundcpy WrongSizeParam\Path 9:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=24>

Status New

The size of the buffer used by cJSON\_strdup in len, at line 49 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that cJSON\_strdup passes to len, at line 49 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	56	56
Object	len	len

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char\* cJSON\_strdup(const char\* str)

```
....
56.             memcpy(copy,str,len);
```

### Buffer Overflow boundcpy WrongSizeParam\Path 10:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=25>

Status New

The size of the buffer used by ensure in p, at line 121 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that ensure passes to p, at line 121 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	131	131

Object	p	p
--------	---	---

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char\* ensure(printbuffer \*p,int needed)

```
....
131.          if (newbuffer) memcpy(newbuffer,p->buffer,p->length);
```

### Buffer Overflow boundcpy WrongSizeParam\Path 11:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=26>

Status New

The size of the buffer used by \*print\_array in tmpen, at line 435 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_array passes to tmpen, at line 435 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	504	504
Object	tmpen	tmpen

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_array(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....
504.          tmpen=strlen(entries[i]);memcpy(ptr,entries[i],tmpen);ptr+=tmpen;
n;
```

### Buffer Overflow boundcpy WrongSizeParam\Path 12:

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=27>

Status New

The size of the buffer used by \*print\_object in tmpen, at line 551 of EasyLogger/cJSON.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that \*print\_object passes to tmpen, at line 551 of EasyLogger/cJSON.c, to overwrite the target buffer.

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	646	646

Object	tmplen	tmplen
--------	--------	--------

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....
646.         tmpflen=strlen(names[i]);memcpy(ptr,names[i],tmpflen);ptr+=tmpflen;
```

## 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=1050077&projectid=50067&pathid=56>

Status New

The variable declared in pvReturn at EasyLogger/heap\_4.c in line 115 is not initialized when it is used by pvReturn at EasyLogger/heap\_4.c in line 115.

	Source	Destination
File	EasyLogger/heap_4.c	EasyLogger/heap_4.c
Line	118	261
Object	pvReturn	pvReturn

#### Code Snippet

File Name EasyLogger/heap\_4.c

Method void \*pvPortMalloc( size\_t xWantedSize )

```
....
118. void *pvReturn = NULL;
....
261.         configASSERT( ( ( ( size_t ) pvReturn ) & ( size_t )
portBYTE_ALIGNMENT_MASK ) == 0 );
```

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

Severity Medium

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=57>

Status New

The variable declared in `pxNextFreeBlock` at `EasyLogger/heap_4.c` in line 333 is not initialized when it is used by `pxNextFreeBlock` at `EasyLogger/heap_4.c` in line 333.

	Source	Destination
File	EasyLogger/heap_4.c	EasyLogger/heap_4.c
Line	364	370
Object	pxNextFreeBlock	pxNextFreeBlock

#### Code Snippet

File Name EasyLogger/heap\_4.c  
Method static void prvHeapInit( void )

```
....  
364.         pxEnd->pxNextFreeBlock = NULL;  
....  
370.         pxFirstFreeBlock->pxNextFreeBlock = pxEnd;
```

## 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	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=4">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=4</a>
Status	New

The `*print_number` method calls the `sprintf` function, at line 147 of `EasyLogger/cJSON.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	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	161	161
Object	sprintf	sprintf

#### Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_number(cJSON \*item, printbuffer \*p)

```
....  
161.         if (str)         sprintf(str, "%d", item->valueint);
```

### Unchecked Return Value\Path 2:

Severity	Low
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=5">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=5</a>
Status	New

The `*print_number` method calls the `sprintf` function, at line 147 of `EasyLogger/cJSON.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	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	169	169
Object	<code>sprintf</code>	<code>sprintf</code>

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_number(cJSON \*item, printbuffer \*p)

```
....  
169.                                if (fpclassify(d) != FP_ZERO && !isnormal(d))  
                                sprintf(str, "null");
```

### Unchecked Return Value\Path 3:

Severity	Low
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=6">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=6</a>
Status	New

The `*print_number` method calls the `sprintf` function, at line 147 of `EasyLogger/cJSON.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	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	170	170
Object	<code>sprintf</code>	<code>sprintf</code>

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_number(cJSON \*item, printbuffer \*p)

```
.....
170.                                else if (fabs(floor(d)-d)<=DBL_EPSILON &&
fabs(d)<1.0e60)    sprintf(str,"%0f",d);
```

#### Unchecked Return Value\Path 4:

Severity	Low
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=7">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=7</a>
Status	New

The \*print\_number method calls the sprintf function, at line 147 of EasyLogger/cJSON.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	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	171	171
Object	sprintf	sprintf

#### Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_number(cJSON \*item, printbuffer \*p)

```
.....
171.                                else if (fabs(d)<1.0e-6 || fabs(d)>1.0e9)
                                sprintf(str,"%e",d);
```

#### Unchecked Return Value\Path 5:

Severity	Low
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=8">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=8</a>
Status	New

The \*print\_number method calls the sprintf function, at line 147 of EasyLogger/cJSON.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	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	172	172
Object	sprintf	sprintf

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_number(cJSON \*item, printbuffer \*p)

```
....
172.                                     else
                                     sprintf(str, "%f", d);
```

#### Unchecked Return Value\Path 6:

Severity	Low
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=9">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=9</a>
Status	New

The \*print\_string\_ptr method calls the sprintf function, at line 253 of EasyLogger/cJSON.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	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	302	302
Object	sprintf	sprintf

#### Code Snippet

File Name EasyLogger/cJSON.c  
Method static char \*print\_string\_ptr(const char \*str, printbuffer \*p)

```
....
302.                                     default:
sprintf(ptr2, "u%04x", token); ptr2+=5;          break;          /* escape and
print */
```

## 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	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=10">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=10</a>
Status	New

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	473	473
Object	sizeof	sizeof

**Code Snippet**

File Name EasyLogger/cJSON.c

Method static char \*print\_array(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
473.      entries=(char**) cJSON_malloc(numentries*sizeof(char));
```

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

Severity Low

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=11>

Status New

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	475	475
Object	sizeof	sizeof

**Code Snippet**

File Name EasyLogger/cJSON.c

Method static char \*print\_array(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
475.      memset(entries,0,numentries*sizeof(char));
```

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

Severity Low

Result State To Verify

Online Results <http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=12>

Status New

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	612	612
Object	sizeof	sizeof

**Code Snippet**

File Name EasyLogger/cJSON.c

Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
612.      entries=(char**) cJSON_malloc(numentries*sizeof(char));
```



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

Severity	Low
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=13">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=13</a>
Status	New

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	614	614
Object	sizeof	sizeof

**Code Snippet**

File Name EasyLogger/cJSON.c

Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
614.          names=(char**) cJSON_malloc(numentries*sizeof(char));
```

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

Severity	Low
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=14">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=14</a>
Status	New

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	616	616
Object	sizeof	sizeof

**Code Snippet**

File Name EasyLogger/cJSON.c

Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....  
616.          memset(entries,0,sizeof(char*)*numentries);
```

**Use of Sizeof On a Pointer Type\Path 6:**

Severity	Low
Result State	To Verify
Online Results	<a href="http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=15">http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&amp;projectid=50067&amp;pathid=15</a>
Status	New

	Source	Destination
File	EasyLogger/cJSON.c	EasyLogger/cJSON.c
Line	617	617
Object	sizeof	sizeof

#### Code Snippet

File Name EasyLogger/cJSON.c

Method static char \*print\_object(cJSON \*item,int depth,int fmt,printbuffer \*p)

```
....
617.          memset (names,0,sizeof (char*) *numentries);
```

## 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-](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=28)

[BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=28](http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050077&projectid=50067&pathid=28)

Status New

The variable declared in 0 at EasyLogger/heap\_4.c in line 333 is not initialized when it is used by xStart at EasyLogger/heap\_4.c in line 333.

	Source	Destination
File	EasyLogger/heap_4.c	EasyLogger/heap_4.c
Line	355	355
Object	0	xStart

#### Code Snippet

File Name EasyLogger/heap\_4.c

Method static void prvHeapInit( void )

```
....
355.          xStart.xBlockSize = ( size_t ) 0;
```

## Buffer Overflow StrcpyStrcat

### 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 its 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 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 its 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 (strlen(inputString, MAX_INPUT_SIZE) < sizeof(buffer))
    {
        strncpy(buffer, inputString, sizeof(buffer));
    }
}
```

# Dangerous Functions

## Risk

### What might happen

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

---

## Cause

### How does it happen

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

---

## General Recommendations

### How to avoid it

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

## Source Code Examples

### CPP

#### Buffer Overflow in gets()

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

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

## Safe reading from user

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

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

## Unsafe function for string copy

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

    return 0;
}
```

## Safe string copy

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

    return 0;
}
```

## Unsafe format string

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

## Safe format string

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



# Use of 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());
```

# 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 function. These functions often return error codes in the result, or share other status codes with its 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

```
buff = (char*) malloc(size);  
strncpy(buff, source, size);
```

#### Safer Memory Allocation

```
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 (strcmp(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 (! strcmp(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 `strcmp()` 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	<a href="#">Pointer Issues</a>	<b>Development Concepts (primary)699</b>
ChildOf	Weakness Class	682	<a href="#">Incorrect Calculation</a>	<b>Research Concepts (primary)1000</b>
ChildOf	Category	737	<a href="#">CERT C Secure Coding Section 03 - Expressions (EXP)</a>	<b>Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734</b>
ChildOf	Category	740	<a href="#">CERT C Secure Coding Section 06 - Arrays (ARR)</a>	Weaknesses Addressed by the CERT C Secure Coding Standard734
CanPrecede	Weakness Base	131	<a href="#">Incorrect Calculation of Buffer Size</a>	Research Concepts1000

## Taxonomy Mappings

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

## White Box Definitions

A weakness where code path has:

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

## References

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

## Content History

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

[BACK TO TOP](#)

# NULL Pointer Dereference

## Risk

### What might happen

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

---

## Cause

### How does it happen

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

---

## General Recommendations

### How to avoid it

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

## Source Code Examples

## Scanned Languages

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