amazon-freertos Scan Report

Project Name	amazon-freertos
Scan Start	Saturday, June 22, 2024 12:05:45 AM
Preset	Checkmarx Default
Scan Time	00h:02m:32s
Lines Of Code Scanned	23487
Files Scanned	22
Report Creation Time	Saturday, June 22, 2024 12:13:02 AM
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074
Team	CxServer
Checkmarx Version	8.7.0
Scan Type	Full
Source Origin	LocalPath
Density	3/1000 (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

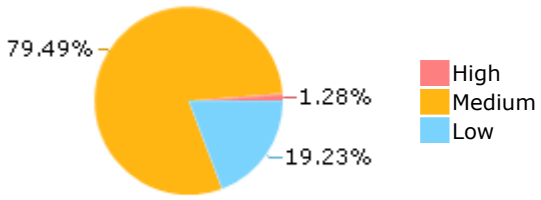
Results Limit

Results limit per query was set to 50

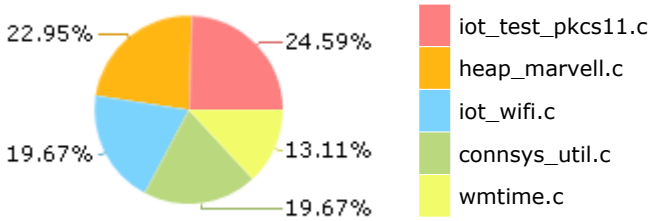
Selected Queries

Selected queries are listed in Result Summary

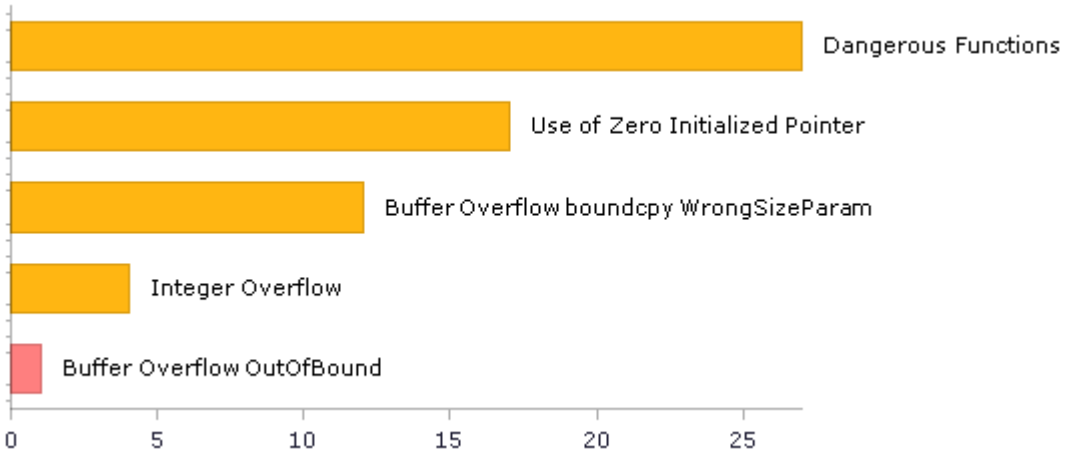
Result Summary



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	23	21
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	1	1
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	1	1
PCI DSS (3.2) - 6.5.2 - Buffer overflows	17	17
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.	1	1
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.	4	4

***** 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)	1	1
SC-4 Information in Shared Resources (P1)	0	0
SC-5 Denial of Service Protection (P1)*	27	20
SC-8 Transmission Confidentiality and Integrity (P1)	0	0
SI-10 Information Input Validation (P1)*	7	7
SI-11 Error Handling (P2)*	2	2
SI-15 Information Output Filtering (P0)	0	0
SI-16 Memory Protection (P1)	1	1

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

Scan Summary - OWASP Mobile Top 10 2016

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

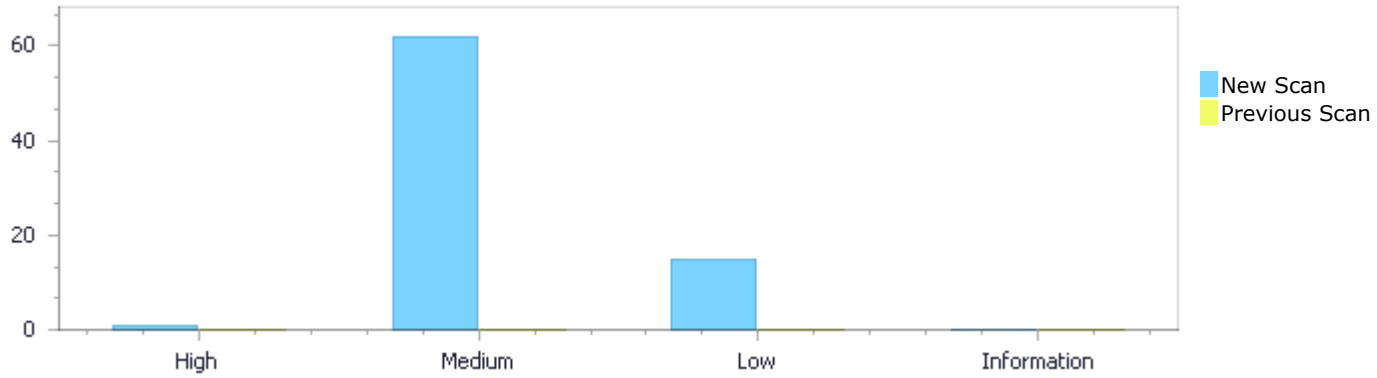
	modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.		
M9-Reverse Engineering	This category includes analysis of the final core binary to determine its source code, libraries, algorithms,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

Scan Summary - Custom

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

Results Distribution By Status First scan of the project

	High	Medium	Low	Information	Total
New Issues	1	62	15	0	78
Recurrent Issues	0	0	0	0	0
Total	1	62	15	0	78
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	1	62	15	0	78
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	1	62	15	0	78

Result Summary

Vulnerability Type	Occurrences	Severity
Buffer Overflow OutOfBound	1	High
Dangerous Functions	27	Medium
Use of Zero Initialized Pointer	17	Medium
Buffer Overflow boundcpy WrongSizeParam	12	Medium
Integer Overflow	4	Medium

Memory Leak	1	Medium
Use After Free	1	Medium
NULL Pointer Dereference	8	Low
Unchecked Array Index	2	Low
Unchecked Return Value	2	Low
Potential Off by One Error in Loops	1	Low
Sizeof Pointer Argument	1	Low
Use of Insufficiently Random Values	1	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
amazon-freertos/iot_test_pkcs11.c	12
amazon-freertos/heap_marvell.c	11
amazon-freertos/iot_wifi.c	10
amazon-freertos/connsys_util.c	10
amazon-freertos/wmtime.c	7
amazon-freertos/sflash_write.c	5
amazon-freertos/heap_3.c	2
amazon-freertos/heap_4.c	2
amazon-freertos/fsl_str.c	1
amazon-freertos/heap_1.c	1

Scan Results Details

Buffer Overflow OutOfBound

Query Path:
CPP\Cx\CPP Buffer Overflow\Buffer Overflow OutOfBound 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 OutOfBound\Path 1:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=1
Status	New

The size of the buffer used by mktime in i, at line 329 of amazon-freertos/wmtime.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that }; passes to SEC_PER_YR, at line 42 of amazon-freertos/wmtime.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/wmtime.c	amazon-freertos/wmtime.c
Line	42	337
Object	SEC_PER_YR	i

Code Snippet
File Name amazon-freertos/wmtime.c
Method uint32_t SEC_PER_YR[2] = { 31536000, 31622400 };

```
....
42.  uint32_t SEC_PER_YR[2] = { 31536000, 31622400 };
```

File Name amazon-freertos/wmtime.c
Method time_t mktime(struct tm *tm)

```
....
337.              ret += SEC_PER_YR[is_leap(i)];
```

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=1050084&projectid=50074&pathid=33
Status	New

The dangerous function, memcpy, was found in use at line 785 in amazon-freertos/heap_marvell.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	828	828
Object	memcpy	memcpy

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method void* pvPortReAlloc(void *pv, size_t xWantedSize)

```
....
828.                            memcpy( newArea, pv, copySize );
```

Dangerous Functions\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=34
Status	New

The dangerous function, memcpy, was found in use at line 1105 in amazon-freertos/iot_test_pkcs11.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	1113	1113
Object	memcpy	memcpy

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method static void prvGenerateRandomMultiThreadTask(void * pvParameters)

```
....
1113.       memcpy( &xSession, pxMultiTaskParam->pvTaskData, sizeof(
CK_SESSION_HANDLE ) );
```

Dangerous Functions\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=35
Status	New

The dangerous function, memcpy, was found in use at line 2249 in amazon-freertos/iot_test_pkcs11.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	2257	2257
Object	memcpy	memcpy

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method static void prvFindObjectMultiThreadTask(void * pvParameters)

```
....
2257.      memcpy( &xSession, pxMultiTaskParam->pvTaskData, sizeof(
CK_SESSION_HANDLE ) );
```

Dangerous Functions\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=36
Status	New

The dangerous function, memcpy, was found in use at line 2375 in amazon-freertos/iot_test_pkcs11.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	2390	2390
Object	memcpy	memcpy

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method static void prvECGetAttributeValueMultiThreadTask(void * pvParameters)

```
....
2390.      memcpy( &xSession, pxMultiTaskParam->pvTaskData, sizeof(
CK_SESSION_HANDLE ) );
```

Dangerous Functions\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=37
Status	New

The dangerous function, memcpy, was found in use at line 149 in amazon-freertos/iot_wifi.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	187	187
Object	memcpy	memcpy

Code Snippet
File Name amazon-freertos/iot_wifi.c
Method WIFIReturnCode_t WIFI_ConnectAP(const WIFINetworkParams_t * const pxNetworkParams)

```
....
187.      memcpy(pcSSID, pxNetworkParams->ucSSID, pxNetworkParams->ucSSIDLength);
```

Dangerous Functions\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=38
Status	New

The dangerous function, memcpy, was found in use at line 149 in amazon-freertos/iot_wifi.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	193	193
Object	memcpy	memcpy

Code Snippet
File Name amazon-freertos/iot_wifi.c
Method WIFIReturnCode_t WIFI_ConnectAP(const WIFINetworkParams_t * const pxNetworkParams)

```
....
193.      memcpy(pcPassword, pxNetworkParams->xPassword.xWPA.cPassphrase, pxNetworkParams->xPassword.xWPA.ucLength);
```

Dangerous Functions\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=39
Status	New

The dangerous function, memcpy, was found in use at line 300 in amazon-freertos/iot_wifi.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	325	325
Object	memcpy	memcpy

Code Snippet
File Name amazon-freertos/iot_wifi.c
Method WIFIReturnCode_t WIFI_Scan(WIFIScanResult_t * pxBuffer,

```
....
325.              memcpy(pxBuffer[i].ucSSID, APs[i].ssid,
wificonfigMAX_SSID_LEN);
```

Dangerous Functions\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=40
Status	New

The dangerous function, memcpy, was found in use at line 300 in amazon-freertos/iot_wifi.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	326	326
Object	memcpy	memcpy

Code Snippet
File Name amazon-freertos/iot_wifi.c
Method WIFIReturnCode_t WIFI_Scan(WIFIScanResult_t * pxBuffer,

```
....
326.              memcpy(pxBuffer[i].ucBSSID, APs[i].mac.mac,
wificonfigMAX_BSSID_LEN);
```

Dangerous Functions\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=41
Status	New

The dangerous function, memcpy, was found in use at line 212 in amazon-freertos/sflash_write.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/sflash_write.c	amazon-freertos/sflash_write.c
Line	505	505
Object	memcpy	memcpy

Code Snippet
File Name amazon-freertos/sflash_write.c
Method int main(void)

```
....
505.                    memcpy( &Rx_Buffer[compare_start], cmp_ptr,
cmp_len );
```

Dangerous Functions\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=42
Status	New

The dangerous function, sprintf, was found in use at line 438 in amazon-freertos/iot_wifi.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	452	452
Object	sprintf	sprintf

Code Snippet
File Name amazon-freertos/iot_wifi.c
Method WIFIReturnCode_t WIFI_Ping(uint8_t * pucIPAddr,

```
....
452.    sprintf(host_name, "%d.%d.%d.%d", pucIPAddr[0], pucIPAddr[1],
pucIPAddr[2], pucIPAddr[3]);
```

Dangerous Functions\Path 11:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=43
Status	New

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

	Source	Destination
File	amazon-freertos/fsl_str.c	amazon-freertos/fsl_str.c
Line	827	827
Object	strlen	strlen

Code Snippet
File Name amazon-freertos/fsl_str.c
Method int StrFormatPrintf(const char *fmt, va_list ap, char *buf, printfCb cb)

```
....
827.                            vlen = strlen(sval);
```

Dangerous Functions\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=44
Status	New

The dangerous function, strlen, was found in use at line 424 in amazon-freertos/iot_test_pkcs11.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	466	466
Object	strlen	strlen

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method static CK_RV prvDestroyTestCredentials(void)

```
....
466.                                                    strlen( ( char * )
pxPkcsLabels[ ulLabelCount ] ),
```

Dangerous Functions\Path 13:

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

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=45
Status	New

The dangerous function, strlen, was found in use at line 110 in amazon-freertos/wmtime.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/wmtime.c	amazon-freertos/wmtime.c
Line	126	126
Object	strlen	strlen

Code Snippet
File Name amazon-freertos/wmtime.c
Method time_t http_date_to_time(const unsigned char *date)

```
....
126.              if (strlen((char *)date) != 29)
```

Dangerous Functions\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=46
Status	New

The dangerous function, strlen, was found in use at line 110 in amazon-freertos/wmtime.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/wmtime.c	amazon-freertos/wmtime.c
Line	182	182
Object	strlen	strlen

Code Snippet
File Name amazon-freertos/wmtime.c
Method time_t http_date_to_time(const unsigned char *date)

```
....
182.              if (strlen((char *)date) != 24)
```

Dangerous Functions\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=47
Status	New

The dangerous function, strlen, was found in use at line 110 in amazon-freertos/wmtime.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/wmtime.c	amazon-freertos/wmtime.c
Line	237	237
Object	strlen	strlen

Code Snippet
File Name amazon-freertos/wmtime.c
Method time_t http_date_to_time(const unsigned char *date)

```
....
237.              if (strlen((char *)date) < 11)
```

Dangerous Functions\Path 16:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=48
Status New

The dangerous function, strlen, was found in use at line 110 in amazon-freertos/wmtime.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/wmtime.c	amazon-freertos/wmtime.c
Line	250	250
Object	strlen	strlen

Code Snippet
File Name amazon-freertos/wmtime.c
Method time_t http_date_to_time(const unsigned char *date)

```
....
250.              if (strlen((char *)start_date) != 22)
```

Dangerous Functions\Path 17:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=49
Status New

The dangerous function, strncpy, was found in use at line 149 in amazon-freertos/iot_wifi.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	196	196
Object	strncpy	strncpy

Code Snippet
File Name amazon-freertos/iot_wifi.c
Method WIFIReturnCode_t WIFI_ConnectAP(const WIFINetworkParams_t * const pxNetworkParams)

```
....
196.      strncpy(pcLastAttemptedSSID, (char *)pxNetworkParams->ucSSID,
pxNetworkParams->ucSSIDLength + 1);
```

Dangerous Functions\Path 18:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=50
Status New

The dangerous function, atoi, was found in use at line 2588 in amazon-freertos/connsys_util.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	2599	2599
Object	atoi	atoi

Code Snippet
File Name amazon-freertos/connsys_util.c
Method uint8_t connsys_util_cli_handler(uint8_t len, char *param[])

```
....
2599.      switch (atoi(param[0])) {
```

Dangerous Functions\Path 19:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=51
Status New

The dangerous function, atoi, was found in use at line 2588 in amazon-freertos/connsys_util.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	2607	2607
Object	atoi	atoi

Code Snippet
File Name amazon-freertos/connsys_util.c
Method uint8_t connsys_util_cli_handler(uint8_t len, char *param[])

```
....
2607.               debug_flag = atoi(param[1]);
```

Dangerous Functions\Path 20:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=52
Status	New

The dangerous function, atoi, was found in use at line 2588 in amazon-freertos/connsys_util.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	2618	2618
Object	atoi	atoi

Code Snippet
File Name amazon-freertos/connsys_util.c
Method uint8_t connsys_util_cli_handler(uint8_t len, char *param[])

```
....
2618.               debug_flag = atoi(param[1]);
```

Dangerous Functions\Path 21:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=53
Status	New

The dangerous function, atoi, was found in use at line 2588 in amazon-freertos/connsys_util.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c

Line	2672	2672
Object	atoi	atoi

Code Snippet
File Name amazon-freertos/connsys_util.c
Method uint8_t connsys_util_cli_handler(uint8_t len, char *param[])

```
....
2672.              config = (uint8_t)atoi(param[1]);
```

Dangerous Functions\Path 22:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=54
Status	New

The dangerous function, atoi, was found in use at line 2588 in amazon-freertos/connsys_util.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	2693	2693
Object	atoi	atoi

Code Snippet
File Name amazon-freertos/connsys_util.c
Method uint8_t connsys_util_cli_handler(uint8_t len, char *param[])

```
....
2693.              reserve_page_num = (uint32_t)atoi(param[1]);
```

Dangerous Functions\Path 23:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=55
Status	New

The dangerous function, atoi, was found in use at line 2588 in amazon-freertos/connsys_util.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	2711	2711
Object	atoi	atoi

Code Snippet	
File Name	amazon-freertos/connsys_util.c
Method	uint8_t connsys_util_cli_handler(uint8_t len, char *param[])

```
....
2711.              config = (uint32_t)atoi(param[1]);
```

Dangerous Functions\Path 24:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=56
Status	New

The dangerous function, atoi, was found in use at line 2588 in amazon-freertos/connsys_util.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	2723	2723
Object	atoi	atoi

Code Snippet	
File Name	amazon-freertos/connsys_util.c
Method	uint8_t connsys_util_cli_handler(uint8_t len, char *param[])

```
....
2723.              num = (uint32_t)atoi(param[1]);
```

Dangerous Functions\Path 25:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=57
Status	New

The dangerous function, atoi, was found in use at line 2588 in amazon-freertos/connsys_util.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	2734	2734
Object	atoi	atoi

Code Snippet	
File Name	amazon-freertos/connsys_util.c

Method	uint8_t connsys_util_cli_handler(uint8_t len, char *param[])

```
....
2734.              num = (uint32_t)atoi(param[1]);
```

Dangerous Functions\Path 26:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=58
Status	New

The dangerous function, atoi, was found in use at line 2588 in amazon-freertos/connsys_util.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	2757	2757
Object	atoi	atoi

Code Snippet	
File Name	amazon-freertos/connsys_util.c
Method	uint8_t connsys_util_cli_handler(uint8_t len, char *param[])

```
....
2757.              offset = (uint32_t)atoi(param[1]);
```

Dangerous Functions\Path 27:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=59
Status	New

The dangerous function, atoi, was found in use at line 2588 in amazon-freertos/connsys_util.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	2764	2764
Object	atoi	atoi

Code Snippet	
File Name	amazon-freertos/connsys_util.c
Method	uint8_t connsys_util_cli_handler(uint8_t len, char *param[])

```
....
2764.              offset = (uint32_t)atoi(param[1]);
```

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=1050084&projectid=50074&pathid=62
Status	New

The variable declared in pucAlignedHeap at amazon-freertos/heap_1.c in line 71 is not initialized when it is used by pucAlignedHeap at amazon-freertos/heap_1.c in line 71.

	Source	Destination
File	amazon-freertos/heap_1.c	amazon-freertos/heap_1.c
Line	74	101
Object	pucAlignedHeap	pucAlignedHeap

Code Snippet
File Name amazon-freertos/heap_1.c
Method void *pvPortMalloc(size_t xWantedSize)

```
....
74.  static uint8_t *pucAlignedHeap = NULL;
....
101.                    pvReturn = pucAlignedHeap + xNextFreeByte;
```

Use of Zero Initialized Pointer\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=63
Status	New

The variable declared in pvReturn at amazon-freertos/heap_4.c in line 114 is not initialized when it is used by pvReturn at amazon-freertos/heap_4.c in line 114.

	Source	Destination
File	amazon-freertos/heap_4.c	amazon-freertos/heap_4.c

Line	117	259
Object	pvReturn	pvReturn

Code Snippet
File Name amazon-freertos/heap_4.c
Method void *pvPortMalloc(size_t xWantedSize)

```
....
117.  void *pvReturn = NULL;
....
259.        configASSERT( ( ( ( size_t ) pvReturn ) & ( size_t )
portBYTE_ALIGNMENT_MASK ) == 0 );
```

Use of Zero Initialized Pointer\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=64
Status	New

The variable declared in pxBlock at amazon-freertos/heap_marvell.c in line 598 is not initialized when it is used by pxBlock at amazon-freertos/heap_marvell.c in line 598.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	600	723
Object	pxBlock	pxBlock

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method void *pvPortMalloc(size_t xWantedSize)

```
....
600.        xBlockLink *pxBlock = NULL, *pxPreviousBlock,
*pxNewBlockLink;
....
723.                    BLOCK_SIZE( pxBlock ),
```

Use of Zero Initialized Pointer\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=65
Status	New

The variable declared in pxPrev at amazon-freertos/heap_marvell.c in line 423 is not initialized when it is used by pxNextFreeBlock at amazon-freertos/heap_marvell.c in line 290.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	441	402
Object	pxPrev	pxNextFreeBlock

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method void prvHeapInit()

```
....
441.        xStart.pxPrev = NULL;
```

File Name amazon-freertos/heap_marvell.c
Method static inline void prvInsertBlockIntoFreeList(xBlockLink * pxBlockToInsert)

```
....
402.        pxBlockToInsert->pxNextFreeBlock = pxIterator-
>pxNextFreeBlock;
```

Use of Zero Initialized Pointer\Path 5:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=66
Status New

The variable declared in pxFunctionList at amazon-freertos/iot_test_pkcs11.c in line 667 is not initialized when it is used by pxFunctionList at amazon-freertos/iot_test_pkcs11.c in line 667.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	669	696
Object	pxFunctionList	pxFunctionList

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method TEST(Full_PKCS11_StartFinish, AFQP_InitializeFinalize)

```
....
669.      CK_FUNCTION_LIST_PTR pxFunctionList = NULL;
....
696.      xResult = pxFunctionList->C_Finalize( NULL );
```

Use of Zero Initialized Pointer\Path 6:

Severity Medium
Result State To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=67
Status	New

The variable declared in pxFunctionList at amazon-freertos/iot_test_pkcs11.c in line 667 is not initialized when it is used by pxFunctionList at amazon-freertos/iot_test_pkcs11.c in line 667.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	669	691
Object	pxFunctionList	pxFunctionList

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method TEST(Full_PKCS11_StartFinish, AFQP_InitializeFinalize)

```
....
669.      CK_FUNCTION_LIST_PTR pxFunctionList = NULL;
....
691.      xResult = pxFunctionList->C_Finalize( NULL );
```

Use of Zero Initialized Pointer\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=68
Status	New

The variable declared in pxFunctionList at amazon-freertos/iot_test_pkcs11.c in line 667 is not initialized when it is used by pxFunctionList at amazon-freertos/iot_test_pkcs11.c in line 667.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	669	687
Object	pxFunctionList	pxFunctionList

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method TEST(Full_PKCS11_StartFinish, AFQP_InitializeFinalize)

```
....
669.      CK_FUNCTION_LIST_PTR pxFunctionList = NULL;
....
687.          xResult = pxFunctionList->C_Finalize( ( CK_VOID_PTR )
0x1234 );
```

Use of Zero Initialized Pointer\Path 8:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=69
Status	New

The variable declared in pxSlotId at amazon-freertos/iot_test_pkcs11.c in line 756 is not initialized when it is used by pxSlotId at amazon-freertos/iot_test_pkcs11.c in line 756.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	758	773
Object	pxSlotId	pxSlotId

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method TEST(Full_PKCS11_StartFinish, AFQP_OpenSessionCloseSession)

```
....
758.      CK_SLOT_ID_PTR pxSlotId = NULL;
....
773.          xSlotId = pxSlotId[ pkcs11testSLOT_NUMBER ];
```

Use of Zero Initialized Pointer\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=70
Status	New

The variable declared in pxNextFreeBlock at amazon-freertos/heap_2.c in line 250 is not initialized when it is used by pxNextFreeBlock at amazon-freertos/heap_2.c in line 250.

	Source	Destination
File	amazon-freertos/heap_2.c	amazon-freertos/heap_2.c
Line	265	271
Object	pxNextFreeBlock	pxNextFreeBlock

Code Snippet
File Name amazon-freertos/heap_2.c
Method static void prvHeapInit(void)

```
....
265.        xEnd.pxNextFreeBlock = NULL;
....
271.        pxFirstFreeBlock->pxNextFreeBlock = &xEnd;
```

Use of Zero Initialized Pointer\Path 10:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=71
Status	New

The variable declared in pxNextFreeBlock at amazon-freertos/heap_4.c in line 330 is not initialized when it is used by pxNextFreeBlock at amazon-freertos/heap_4.c in line 330.

	Source	Destination
File	amazon-freertos/heap_4.c	amazon-freertos/heap_4.c
Line	361	367
Object	pxNextFreeBlock	pxNextFreeBlock

Code Snippet
File Name amazon-freertos/heap_4.c
Method static void prvHeapInit(void)

```
....
361.        pxEnd->pxNextFreeBlock = NULL;
....
367.        pxFirstFreeBlock->pxNextFreeBlock = pxEnd;
```

Use of Zero Initialized Pointer\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=72
Status	New

The variable declared in pxNextFreeBlock at amazon-freertos/heap_5.c in line 394 is not initialized when it is used by pxNextFreeBlock at amazon-freertos/heap_5.c in line 394.

	Source	Destination
File	amazon-freertos/heap_5.c	amazon-freertos/heap_5.c
Line	454	461
Object	pxNextFreeBlock	pxNextFreeBlock

Code Snippet
File Name amazon-freertos/heap_5.c
Method void vPortDefineHeapRegions(const HeapRegion_t * const pxHeapRegions)

```
....
454.            pxEnd->pxNextFreeBlock = NULL;
....
461.            pxFirstFreeBlockInRegion->pxNextFreeBlock = pxEnd;
```

Use of Zero Initialized Pointer\Path 12:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=73
Status	New

The variable declared in pxPrev at amazon-freertos/heap_marvell.c in line 423 is not initialized when it is used by pxNextFreeBlock at amazon-freertos/heap_marvell.c in line 598.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	441	650
Object	pxPrev	pxNextFreeBlock

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method void prvHeapInit()

```
....
441.        xStart.pxPrev = NULL;
```

File Name amazon-freertos/heap_marvell.c
Method void *pvPortMalloc(size_t xWantedSize)

```
....
650.                        pvReturn = ( void * ) ( ( ( unsigned char
* ) pxPreviousBlock->pxNextFreeBlock )
```

Use of Zero Initialized Pointer\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=74
Status	New

The variable declared in pxPrev at amazon-freertos/heap_marvell.c in line 423 is not initialized when it is used by pxNextFreeBlock at amazon-freertos/heap_marvell.c in line 467.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	446	570
Object	pxPrev	pxNextFreeBlock

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method void prvHeapInit()

```
....
446.        xEnd.pxPrev = NULL;
```

File Name amazon-freertos/heap_marvell.c

Method int prvHeapAddMemBank(char *chunk_start, size_t size)

```
....
570.        pxNewBlock->pxNextFreeBlock = &xEnd;
```

Use of Zero Initialized Pointer\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=75
Status	New

The variable declared in pxNextFreeBlock at amazon-freertos/heap_marvell.c in line 423 is not initialized when it is used by pxNextFreeBlock at amazon-freertos/heap_marvell.c in line 467.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	447	570
Object	pxNextFreeBlock	pxNextFreeBlock

Code Snippet

File Name amazon-freertos/heap_marvell.c

Method void prvHeapInit()

```
....
447.        xEnd.pxNextFreeBlock = NULL;
```

File Name amazon-freertos/heap_marvell.c

Method int prvHeapAddMemBank(char *chunk_start, size_t size)

```
....
570.        pxNewBlock->pxNextFreeBlock = &xEnd;
```

Use of Zero Initialized Pointer\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=76
Status	New

The variable declared in pxPrev at amazon-freertos/heap_marvell.c in line 423 is not initialized when it is used by pxNextFreeBlock at amazon-freertos/heap_marvell.c in line 423.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	446	453
Object	pxPrev	pxNextFreeBlock

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method void prvHeapInit()

```
....
446.        xEnd.pxPrev = NULL;
....
453.        pxFirstFreeBlock->pxNextFreeBlock = &xEnd;
```

Use of Zero Initialized Pointer\Path 16:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=77
Status New

The variable declared in pxNextFreeBlock at amazon-freertos/heap_marvell.c in line 423 is not initialized when it is used by pxNextFreeBlock at amazon-freertos/heap_marvell.c in line 423.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	447	453
Object	pxNextFreeBlock	pxNextFreeBlock

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method void prvHeapInit()

```
....
447.        xEnd.pxNextFreeBlock = NULL;
....
453.        pxFirstFreeBlock->pxNextFreeBlock = &xEnd;
```

Use of Zero Initialized Pointer\Path 17:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=78
Status New

The variable declared in pxNextFreeBlock at amazon-freertos/heap_marvell.c in line 467 is not initialized when it is used by pxPrev at amazon-freertos/heap_marvell.c in line 467.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	552	571
Object	pxNextFreeBlock	pxPrev

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method int prvHeapAddMemBank(char *chunk_start, size_t size)

```
....
552.        pxAllocBlock->pxNextFreeBlock = NULL;
....
571.        pxNewBlock->pxPrev = pxAllocBlock;
```

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=1050084&projectid=50074&pathid=6
Status	New

The size of the buffer used by prvGenerateRandomMultiThreadTask in CK_SESSION_HANDLE, at line 1105 of amazon-freertos/iot_test_pkcs11.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that prvGenerateRandomMultiThreadTask passes to CK_SESSION_HANDLE, at line 1105 of amazon-freertos/iot_test_pkcs11.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	1113	1113
Object	CK_SESSION_HANDLE	CK_SESSION_HANDLE

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method static void prvGenerateRandomMultiThreadTask(void * pvParameters)

```
....
1113.      memcpy( &xSession, pxMultiTaskParam->pvTaskData, sizeof(
CK_SESSION_HANDLE ) );
```

Buffer Overflow boundcpy WrongSizeParam\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=7
Status	New

The size of the buffer used by prvFindObjectMultiThreadTask in CK_SESSION_HANDLE, at line 2249 of amazon-freertos/iot_test_pkcs11.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that prvFindObjectMultiThreadTask passes to CK_SESSION_HANDLE, at line 2249 of amazon-freertos/iot_test_pkcs11.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	2257	2257
Object	CK_SESSION_HANDLE	CK_SESSION_HANDLE

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method static void prvFindObjectMultiThreadTask(void * pvParameters)

```
....
2257.      memcpy( &xSession, pxMultiTaskParam->pvTaskData, sizeof(
CK_SESSION_HANDLE ) );
```

Buffer Overflow boundcpy WrongSizeParam\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=8
Status	New

The size of the buffer used by prvECGetAttributeValueMultiThreadTask in CK_SESSION_HANDLE, at line 2375 of amazon-freertos/iot_test_pkcs11.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that prvECGetAttributeValueMultiThreadTask passes to CK_SESSION_HANDLE, at line 2375 of amazon-freertos/iot_test_pkcs11.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	2390	2390
Object	CK_SESSION_HANDLE	CK_SESSION_HANDLE

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c

Method	static void prvECGetAttributeValueMultiThreadTask(void * pvParameters)

```
....
2390.      memcpy( &xSession, pxMultiTaskParam->pvTaskData, sizeof(
CK_SESSION_HANDLE ) );
```

Buffer Overflow boundcpy WrongSizeParam\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=9
Status	New

The size of the buffer used by prvHeapInit in heapAllocatorInfo_t, at line 423 of amazon-freertos/heap_marvell.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that prvHeapInit passes to heapAllocatorInfo_t, at line 423 of amazon-freertos/heap_marvell.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	460	460
Object	heapAllocatorInfo_t	heapAllocatorInfo_t

Code Snippet	
File Name	amazon-freertos/heap_marvell.c
Method	void prvHeapInit()

```
....
460.        memset(&hI, 0x00, sizeof(heapAllocatorInfo_t));
```

Buffer Overflow boundcpy WrongSizeParam\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=10
Status	New

The size of the buffer used by TEST in ucP256Oid, at line 2077 of amazon-freertos/iot_test_pkcs11.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that TEST passes to ucP256Oid, at line 2077 of amazon-freertos/iot_test_pkcs11.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	2184	2184
Object	ucP256Oid	ucP256Oid

Code Snippet	
File Name	amazon-freertos/iot_test_pkcs11.c

Method	TEST(Full_PKCS11_EC, AFQP_GetAttributeValue)

```
....
2184.      memset( xEcParams, 0x0, sizeof( ucP256Oid ) );
```

Buffer Overflow boundcpy WrongSizeParam\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=11
Status	New

The size of the buffer used by WIFI_GetIPInfo in WIFIIPConfiguration_t, at line 484 of amazon-freertos/iot_wifi.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that WIFI_GetIPInfo passes to WIFIIPConfiguration_t, at line 484 of amazon-freertos/iot_wifi.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	496	496
Object	WIFIIPConfiguration_t	WIFIIPConfiguration_t

Code Snippet	
File Name	amazon-freertos/iot_wifi.c
Method	WIFIReturnCode_t WIFI_GetIPInfo(WIFIIPConfiguration_t * xIPConfig)

```
....
496.    memset(xIPConfig, 0, sizeof(WIFIIPConfiguration_t));
```

Buffer Overflow boundcpy WrongSizeParam\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=12
Status	New

The size of the buffer used by http_date_to_time in tm, at line 110 of amazon-freertos/wmtime.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that http_date_to_time passes to tm, at line 110 of amazon-freertos/wmtime.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/wmtime.c	amazon-freertos/wmtime.c
Line	121	121
Object	tm	tm

Code Snippet	
File Name	amazon-freertos/wmtime.c
Method	time_t http_date_to_time(const unsigned char *date)

```
....
121.        memset(&tm_time, 0, sizeof(struct tm));
```

Buffer Overflow boundcpy WrongSizeParam\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=13
Status	New

The size of the buffer used by *gmtime_r in tm, at line 354 of amazon-freertos/wmtime.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *gmtime_r passes to tm, at line 354 of amazon-freertos/wmtime.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/wmtime.c	amazon-freertos/wmtime.c
Line	359	359
Object	tm	tm

Code Snippet
File Name amazon-freertos/wmtime.c
Method struct tm *gmtime_r(const time_t * time, struct tm *result)

```
....
359.        memset(result, 0, sizeof(struct tm));
```

Buffer Overflow boundcpy WrongSizeParam\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=14
Status	New

The size of the buffer used by pvPortReAlloc in copySize, at line 785 of amazon-freertos/heap_marvell.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pvPortReAlloc passes to copySize, at line 785 of amazon-freertos/heap_marvell.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	828	828
Object	copySize	copySize

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method void* pvPortReAlloc(void *pv, size_t xWantedSize)

```
....
828.                          memcpy( newArea, pv, copySize );
```

Buffer Overflow boundcpy WrongSizeParam\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=15
Status	New

The size of the buffer used by WIFI_ConnectAP in pxNetworkParams, at line 149 of amazon-freertos/iot_wifi.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that WIFI_ConnectAP passes to pxNetworkParams, at line 149 of amazon-freertos/iot_wifi.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	187	187
Object	pxNetworkParams	pxNetworkParams

Code Snippet
File Name amazon-freertos/iot_wifi.c
Method WIFIReturnCode_t WIFI_ConnectAP(const WIFINetworkParams_t * const pxNetworkParams)

```
....
187.      memcpy(pcSSID, pxNetworkParams->ucSSID, pxNetworkParams->ucSSIDLength);
```

Buffer Overflow boundcpy WrongSizeParam\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=16
Status	New

The size of the buffer used by WIFI_ConnectAP in pxNetworkParams, at line 149 of amazon-freertos/iot_wifi.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that WIFI_ConnectAP passes to pxNetworkParams, at line 149 of amazon-freertos/iot_wifi.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	193	193
Object	pxNetworkParams	pxNetworkParams

Code Snippet
File Name amazon-freertos/iot_wifi.c

Method	WIFIReturnCode_t WIFI_ConnectAP(const WIFINetworkParams_t * const pxNetworkParams)

```
....
193.      memcpy(pcPassword, pxNetworkParams-
>xPassword.xWPA.cPassphrase, pxNetworkParams->xPassword.xWPA.ucLength);
```

Buffer Overflow boundcpy WrongSizeParam\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=17
Status	New

The size of the buffer used by WIFI_Scan in wificonfigMAX_BSSID_LEN, at line 300 of amazon-freertos/iot_wifi.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that WIFI_Scan passes to wificonfigMAX_BSSID_LEN, at line 300 of amazon-freertos/iot_wifi.c, to overwrite the target buffer.

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	326	326
Object	wificonfigMAX_BSSID_LEN	wificonfigMAX_BSSID_LEN

Code Snippet	
File Name	amazon-freertos/iot_wifi.c
Method	WIFIReturnCode_t WIFI_Scan(WIFIScanResult_t * pxBuffer,

```
....
326.            memcpy(pxBuffer[i].ucBSSID, APs[i].mac.mac,
wificonfigMAX_BSSID_LEN);
```

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=1050084&projectid=50074&pathid=26
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 212 of amazon-freertos/sflash_write.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	amazon-freertos/sflash_write.c	amazon-freertos/sflash_write.c
Line	387	387
Object	AssignExpr	AssignExpr

Code Snippet
File Name amazon-freertos/sflash_write.c
Method int main(void)

```
....
387.                        read_size = ( chip_size - pos > (unsigned
long) sizeof(Rx_Buffer) )? (unsigned int) sizeof(Rx_Buffer) : (unsigned
int) ( chip_size - pos );
```

Integer Overflow\Path 2:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=27
Status New

A variable of a larger data type, read_size, is being assigned to a smaller data type, in 212 of amazon-freertos/sflash_write.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	amazon-freertos/sflash_write.c	amazon-freertos/sflash_write.c
Line	339	339
Object	read_size	read_size

Code Snippet
File Name amazon-freertos/sflash_write.c
Method int main(void)

```
....
339.                    unsigned int read_size = ( data_transfer.size -
pos > (unsigned long) sizeof(Rx_Buffer) )? (unsigned int)
sizeof(Rx_Buffer) : (unsigned int) ( data_transfer.size - pos );
```

Integer Overflow\Path 3:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=28
Status New

A variable of a larger data type, write_size, is being assigned to a smaller data type, in 212 of amazon-freertos/sflash_write.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	amazon-freertos/sflash_write.c	amazon-freertos/sflash_write.c
Line	422	422
Object	write_size	write_size

Code Snippet
File Name amazon-freertos/sflash_write.c
Method int main(void)

```
....
422.                    unsigned int write_size = ( data_transfer.size -
pos > (unsigned long) WRITE_CHUNK_SIZE )? (unsigned int)
WRITE_CHUNK_SIZE : (unsigned int) ( data_transfer.size - pos );
```

Integer Overflow\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=29
Status	New

A variable of a larger data type, read_size, is being assigned to a smaller data type, in 212 of amazon-freertos/sflash_write.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	amazon-freertos/sflash_write.c	amazon-freertos/sflash_write.c
Line	532	532
Object	read_size	read_size

Code Snippet
File Name amazon-freertos/sflash_write.c
Method int main(void)

```
....
532.                    unsigned int read_size = ( data_transfer.size -
pos > (unsigned long) sizeof(Rx_Buffer) )? (unsigned int)
sizeof(Rx_Buffer) : data_transfer.size - pos;
```

Memory Leak

Query Path:
CPP\Cx\CPP Medium Threat\Memory Leak Version:1

Categories

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

Description

Memory Leak\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=60
Status	New

	Source	Destination
File	amazon-freertos/heap_3.c	amazon-freertos/heap_3.c
Line	65	65
Object	pvReturn	pvReturn

Code Snippet
File Name amazon-freertos/heap_3.c
Method void *pvPortMalloc(size_t xWantedSize)

```
....
65.         pvReturn = malloc( xWantedSize );
```

Use After Free

Query Path:
CPP\Cx\CPP Medium Threat\Use After Free Version:1

Categories

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

Description

Use After Free\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=61
Status	New

The pointer pv at amazon-freertos/heap_3.c in line 84 is being used after it has been freed.

	Source	Destination
File	amazon-freertos/heap_3.c	amazon-freertos/heap_3.c
Line	90	91
Object	pv	pv

Code Snippet
File Name amazon-freertos/heap_3.c
Method void vPortFree(void *pv)

```
....
90.                 free( pv );
91.                 traceFREE( pv, 0 );
```

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=1050084&projectid=50074&pathid=18
Status	New

The variable declared in null at amazon-freertos/connsys_util.c in line 1903 is not initialized when it is used by wifi_event_hdr at amazon-freertos/connsys_util.c in line 1805.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	1906	1827
Object	null	wifi_event_hdr

Code Snippet
File Name amazon-freertos/connsys_util.c
Method void connsys_intr_enhance_mode_receive_one_data(int32_t port, int16_t rx_len)

```
....
1906.       uint8_t *payload_ptr = NULL;
```

File Name amazon-freertos/connsys_util.c
Method void connsys_dispatch(void *pkt, uint8_t *payload, int port, unsigned int len)

```
....
1827.                                                   wifi_event_hdr-
>ucPacketOffset));
```

NULL Pointer Dereference\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=500

	74&pathid=19
Status	New

The variable declared in null at amazon-freertos/heap_marvell.c in line 598 is not initialized when it is used by pxBlock at amazon-freertos/heap_marvell.c in line 598.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	600	723
Object	null	pxBlock

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method void *pvPortMalloc(size_t xWantedSize)

```
....
600.        xBlockLink *pxBlock = NULL, *pxPreviousBlock,
*pxNewBlockLink;
....
723.                    BLOCK_SIZE( pxBlock ),
```

NULL Pointer Dereference\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=20
Status	New

The variable declared in null at amazon-freertos/iot_test_pkcs11.c in line 667 is not initialized when it is used by pxFunctionList at amazon-freertos/iot_test_pkcs11.c in line 667.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	669	691
Object	null	pxFunctionList

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method TEST(Full_PKCS11_StartFinish, AFQP_InitializeFinalize)

```
....
669.      CK_FUNCTION_LIST_PTR pxFunctionList = NULL;
....
691.      xResult = pxFunctionList->C_Finalize( NULL );
```

NULL Pointer Dereference\Path 4:

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

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=21
Status	New

The variable declared in null at amazon-freertos/iot_test_pkcs11.c in line 667 is not initialized when it is used by pxFunctionList at amazon-freertos/iot_test_pkcs11.c in line 667.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	669	696
Object	null	pxFunctionList

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method TEST(Full_PKCS11_StartFinish, AFQP_InitializeFinalize)

```
....
669.      CK_FUNCTION_LIST_PTR pxFunctionList = NULL;
....
696.      xResult = pxFunctionList->C_Finalize( NULL );
```

NULL Pointer Dereference\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=22
Status	New

The variable declared in null at amazon-freertos/iot_test_pkcs11.c in line 667 is not initialized when it is used by pxFunctionList at amazon-freertos/iot_test_pkcs11.c in line 667.

	Source	Destination
File	amazon-freertos/iot_test_pkcs11.c	amazon-freertos/iot_test_pkcs11.c
Line	669	687
Object	null	pxFunctionList

Code Snippet
File Name amazon-freertos/iot_test_pkcs11.c
Method TEST(Full_PKCS11_StartFinish, AFQP_InitializeFinalize)

```
....
669.      CK_FUNCTION_LIST_PTR pxFunctionList = NULL;
....
687.          xResult = pxFunctionList->C_Finalize( ( CK_VOID_PTR )
0x1234 );
```

NULL Pointer Dereference\Path 6:

Severity	Low
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=23
Status	New

The variable declared in 0 at amazon-freertos/heap_2.c in line 250 is not initialized when it is used by xStart at amazon-freertos/heap_2.c in line 250.

	Source	Destination
File	amazon-freertos/heap_2.c	amazon-freertos/heap_2.c
Line	261	261
Object	0	xStart

Code Snippet
File Name amazon-freertos/heap_2.c
Method static void prvHeapInit(void)

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

NULL Pointer Dereference\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=24
Status	New

The variable declared in 0 at amazon-freertos/heap_4.c in line 330 is not initialized when it is used by xStart at amazon-freertos/heap_4.c in line 330.

	Source	Destination
File	amazon-freertos/heap_4.c	amazon-freertos/heap_4.c
Line	352	352
Object	0	xStart

Code Snippet
File Name amazon-freertos/heap_4.c
Method static void prvHeapInit(void)

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

NULL Pointer Dereference\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=25

Status	New

The variable declared in 0 at amazon-freertos/heap_marvell.c in line 423 is not initialized when it is used by xStart at amazon-freertos/heap_marvell.c in line 423.

	Source	Destination
File	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	442	442
Object	0	xStart

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method void prvHeapInit()

```
....
442.        xStart.xBlockSize = ( size_t ) 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=1050084&projectid=50074&pathid=3
Status	New

The WIFI_Ping method calls the sprintf function, at line 438 of amazon-freertos/iot_wifi.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	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	452	452
Object	sprintf	sprintf

Code Snippet
File Name amazon-freertos/iot_wifi.c
Method WIFIReturnCode_t WIFI_Ping(uint8_t * pucIPAddr,

```
....
452.    sprintf(host_name, "%d.%d.%d.%d", pucIPAddr[0], pucIPAddr[1],
pucIPAddr[2], pucIPAddr[3]);
```

Unchecked Return Value\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=4
Status	New

The *asctime method calls the snprintf function, at line 499 of amazon-freertos/wmtime.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	amazon-freertos/wmtime.c	amazon-freertos/wmtime.c
Line	505	505
Object	snprintf	snprintf

Code Snippet
File Name amazon-freertos/wmtime.c
Method char *asctime(const struct tm *tm)

```
....
505.        snprintf(asctime_buf, STD_ASCTIME_BUF_SIZE,
```

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=1050084&projectid=50074&pathid=31
Status	New

	Source	Destination
File	amazon-freertos/fsl_pint.c	amazon-freertos/fsl_pint.c
Line	131	131
Object	intr	intr

Code Snippet
File Name amazon-freertos/fsl_pint.c
Method void PINT_PinInterruptConfig(PINT_Type *base, pint_pin_int_t intr, pint_pin_enable_t enable, pint_cb_t callback)

```
....
131.      s_pintCallback[intr] = callback;
```

Unchecked Array Index\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050084&projectid=50074&pathid=32
Status	New

	Source	Destination
File	amazon-freertos/iot_wifi.c	amazon-freertos/iot_wifi.c
Line	194	194
Object	ucLength	ucLength

Code Snippet
File Name amazon-freertos/iot_wifi.c
Method WIFIReturnCode_t WIFI_ConnectAP(const WIFINetworkParams_t * const pxNetworkParams)

```
....
194.      pcPassword[pxNetworkParams->xPassword.xWPA.ucLength] = '\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=1050084&projectid=50074&pathid=2
Status	New

Method randomizeAreaData at line 247 of amazon-freertos/heap_marvell.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	amazon-freertos/heap_marvell.c	amazon-freertos/heap_marvell.c
Line	252	252

Object	rand	rand

Code Snippet
File Name amazon-freertos/heap_marvell.c
Method static inline void randomizeAreaData(unsigned char *addr, int size)

```
....
252.              addr[i] = (unsigned char)rand();
```

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=1050084&projectid=50074&pathid=5
Status	New

The buffer allocated by <= in amazon-freertos/connsys_util.c at line 2524 does not correctly account for the actual size of the value, resulting in an incorrect allocation that is off by one.

	Source	Destination
File	amazon-freertos/connsys_util.c	amazon-freertos/connsys_util.c
Line	2533	2533
Object	<=	<=

Code Snippet
File Name amazon-freertos/connsys_util.c
Method void connsys_cli_dump_pse_reg(void)

```
....
2533.      for (offset = 0x0; offset <= 0x18; offset += 4) {
```

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=1050084&projectid=500

	74&pathid=30
Status	New

	Source	Destination
File	amazon-freertos/atcacert_date.c	amazon-freertos/atcacert_date.c
Line	1069	1069
Object	ATCACERT_DATE_FORMAT_SIZES	sizeof

Code Snippet
File Name amazon-freertos/atcacert_date.c
Method int atcacert_date_dec_compcert(const uint8_t enc_dates[3],

```
....
1069.       if (enc_dates == NULL || issue_date == NULL || expire_date ==
NULL || expire_date_format < 0 || expire_date_format >=
sizeof(ATCACERT_DATE_FORMAT_SIZES) /
sizeof(ATCACERT_DATE_FORMAT_SIZES[0]))
```

Buffer Overflow OutOfBound

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));
    }
}
```

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

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;
}
```

Failure to Release Memory Before Removing Last Reference ('Memory Leak')

Weakness ID: 401 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The software does not sufficiently track and release allocated memory after it has been used, which slowly consumes remaining memory.

Extended Description

This is often triggered by improper handling of malformed data or unexpectedly interrupted sessions.

Terminology Notes

"memory leak" has sometimes been used to describe other kinds of issues, e.g. for information leaks in which the contents of memory are inadvertently leaked (CVE-2003-0400 is one such example of this terminology conflict).

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

C

C++

Modes of Introduction

Memory leaks have two common and sometimes overlapping causes:

- Error conditions and other exceptional circumstances
- Confusion over which part of the program is responsible for freeing the memory

Common Consequences

Scope	Effect
Availability	Most memory leaks result in general software reliability problems, but if an attacker can intentionally trigger a memory leak, the attacker might be able to launch a denial of service attack (by crashing or hanging the program) or take advantage of other unexpected program behavior resulting from a low memory condition.

Likelihood of Exploit

Medium

Demonstrative Examples

Example 1

The following C function leaks a block of allocated memory if the call to read() fails to return the expected number of bytes:

(Bad Code)

Example Language: **C**

```
char* getBlock(int fd) {
char* buf = (char*) malloc(BLOCK_SIZE);
if (!buf) {
return NULL;
}
if (read(fd, buf, BLOCK_SIZE) != BLOCK_SIZE) {

return NULL;
}
```

```
return buf;
}
```

Example 2

Here the problem is that every time a connection is made, more memory is allocated. So if one just opened up more and more connections, eventually the machine would run out of memory.

(Bad Code)

Example Language: **C**

```
bar connection(){
foo = malloc(1024);
return foo;
}
endConnection(bar foo) {

free(foo);
}
int main() {

while(1) //thread 1
//On a connection
foo=connection(); //thread 2
//When the connection ends
endConnection(foo)
}
```

Observed Examples

Reference	Description
CVE-2005-3119	Memory leak because function does not free() an element of a data structure.
CVE-2004-0427	Memory leak when counter variable is not decremented.
CVE-2002-0574	Memory leak when counter variable is not decremented.
CVE-2005-3181	Kernel uses wrong function to release a data structure, preventing data from being properly tracked by other code.
CVE-2004-0222	Memory leak via unknown manipulations as part of protocol test suite.
CVE-2001-0136	Memory leak via a series of the same command.

Potential Mitigations

Pre-design: Use a language or compiler that performs automatic bounds checking.

Phase: Architecture and Design

Use an abstraction library to abstract away risky APIs. Not a complete solution.

Pre-design through Build: The Boehm-Demers-Weiser Garbage Collector or valgrind can be used to detect leaks in code. This is not a complete solution as it is not 100% effective.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	398	Indicator of Poor Code Quality	**Seven Pernicious Kingdoms (primary)700**
ChildOf	Category	399	Resource Management Errors	**Development Concepts (primary)699**
ChildOf	Category	633	Weaknesses that Affect Memory	**Resource-specific Weaknesses (primary)631**
ChildOf	Category	730	OWASP Top Ten 2004 Category A9 - Denial of Service	**Weaknesses in OWASP Top Ten (2004) (primary)711**
ChildOf	Weakness Base	772	Missing Release of Resource after Effective	**Research Concepts (primary)1000**

			Lifetime	
MemberOf	View	630	Weaknesses Examined by SAMATE	**Weaknesses Examined by SAMATE (primary)630**
CanFollow	Weakness Class	390	Detection of Error Condition Without Action	Research Concepts1000

Relationship Notes

This is often a resultant weakness due to improper handling of malformed data or early termination of sessions.

Affected Resources

- Memory

Functional Areas

- Memory management

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Memory leak
7 Pernicious Kingdoms			Memory Leak
CLASP			Failure to deallocate data
OWASP Top Ten 2004	A9	CWE More Specific	Denial of Service

White Box Definitions

A weakness where the code path has:

1. start statement that allocates dynamically allocated memory resource

2. end statement that loses identity of the dynamically allocated memory resource creating situation where dynamically allocated memory resource is never relinquished

Where "loses" is defined through the following scenarios:

1. identity of the dynamic allocated memory resource never obtained

2. the statement assigns another value to the data element that stored the identity of the dynamically allocated memory resource and there are no aliases of that data element

3. identity of the dynamic allocated memory resource obtained but never passed on to function for memory resource release

4. the data element that stored the identity of the dynamically allocated resource has reached the end of its scope at the statement and there are no aliases of that data element

References

J. Whittaker and H. Thompson. "How to Break Software Security". Addison Wesley. 2003.

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	PLOVER		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		
2008-08-15		Veracode	External
	Suggested OWASP Top Ten 2004 mapping		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Relationships, Other Notes, References, Relationship Notes, Taxonomy Mappings, Terminology Notes		
2008-10-14	CWE Content Team	MITRE	Internal
	updated Description		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Other Notes		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Name		
2009-07-17	KDM Analytics		External
	Improved the White Box Definition		

2009-07-27	CWE Content Team	MITRE	Internal
	updated White Box Definitions		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Modes of Introduction, Other Notes		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Relationships		

Previous Entry Names

Change Date	Previous Entry Name
2008-04-11	Memory Leak
2009-05-27	Failure to Release Memory Before Removing Last Reference (aka 'Memory Leak')

BACK TO TOP

Use After Free

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
//..
```

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

```
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';
```

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

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

Likelihood of Exploit

High

Detection Methods

Automated Static Analysis

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

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

Effectiveness: High

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

Automated Dynamic Analysis

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

Black Box

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

Demonstrative Examples

Example 1

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

(Bad Code)

Example Language: **C**

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

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

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

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

(Good Code)

Example Language: **C**

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

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

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

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

Example 2

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

(Bad Code)

Example Language: **Java**

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

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

Example 3

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

(Bad Code)

Example Language: **Java**

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

String productSummary = new String("");

try {
String productSummary = getProductSummary(index);

} catch (Exception ex) {...}

return productSummary;
}

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

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

(Good Code)

Example Language: **Java**

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

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

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

} catch (Exception ex) {...}

return productSummary;
}

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

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

return productSummary;
}
```

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

(Good Code)

Example Language: **Java**

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

Observed Examples

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

Potential Mitigations

Phase: Architecture and Design

Strategies: Input Validation; Libraries or Frameworks

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

Phase: Architecture and Design

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

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

Phase: Requirements

Strategy: Language Selection

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

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

Phase: Implementation

Strategy: Input Validation

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

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

Phase: Implementation

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

Weakness Ordinalities

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

Relationships

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

Theoretical Notes

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

Affected Resources

- Memory

f Causal Nature

Explicit

Taxonomy Mappings

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

Related Attack Patterns

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

References

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

Content History

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

BACK TO TOP

Scanned Languages

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