ElectronBot Scan Report

Project Name	ElectronBot
Scan Start	Friday, June 21, 2024 11:19:30 PM
Preset	Checkmarx Default
Scan Time	00h:14m:44s
Lines Of Code Scanned	4212
Files Scanned	3
Report Creation Time	Friday, June 21, 2024 11:34:43 PM
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065
Team	CxServer
Checkmarx Version	8.7.0
Scan Type	Full
Source Origin	LocalPath
Density	4/1000 (Vulnerabilities/LOC)
Visibility	Public

Filter Settings

Severity

Included: High, Medium, Low, Information

Excluded: None

Result State

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

Excluded: None

Assigned to

Included: All

Categories

Included:

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

Excluded:

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

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

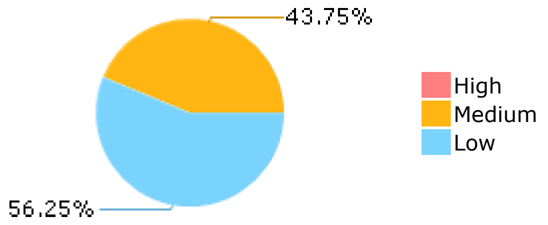
Results Limit

Results limit per query was set to 50

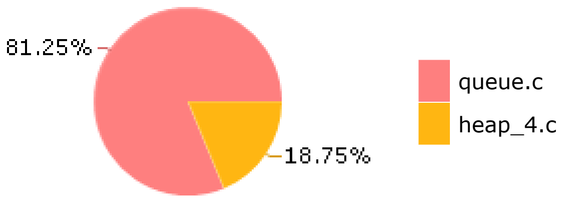
Selected Queries

Selected queries are listed in Result Summary

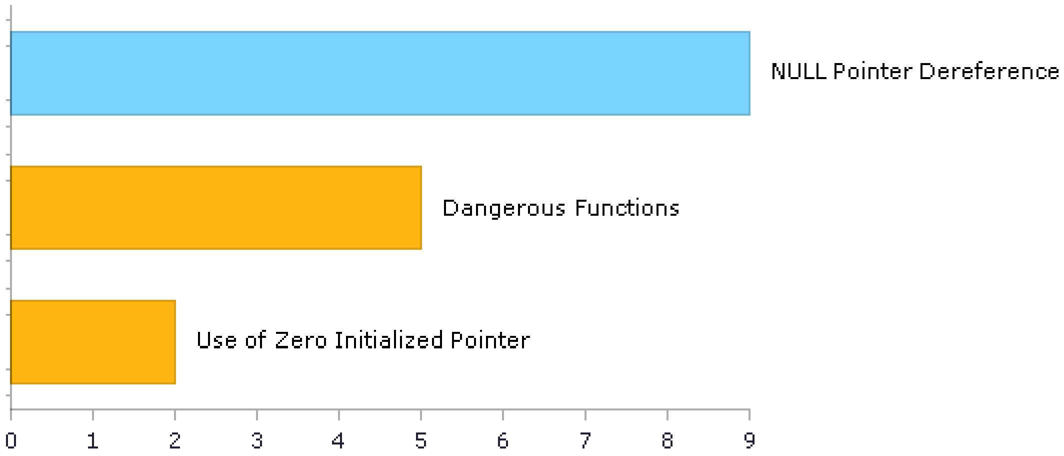
Result Summary



Most Vulnerable Files



Top 5 Vulnerabilities



Scan Summary - OWASP Top 10 2017

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

Category	Threat Agent	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection	App. Specific	EASY	COMMON	EASY	SEVERE	App. Specific	9	4
A2-Broken Authentication	App. Specific	EASY	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A3-Sensitive Data Exposure	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App. Specific	0	0
A4-XML External Entities (XXE)	App. Specific	AVERAGE	COMMON	EASY	SEVERE	App. Specific	0	0
A5-Broken Access Control*	App. Specific	AVERAGE	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A6-Security Misconfiguration	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A7-Cross-Site Scripting (XSS)	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A8-Insecure Deserialization	App. Specific	DIFFICULT	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A9-Using Components with Known Vulnerabilities*	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	MODERATE	App. Specific	5	5
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	5	5
A10-Unvalidated Redirects and Forwards	USERS BROWSERS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0

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

Scan Summary - PCI DSS v3.2

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

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

Scan Summary - FISMA 2014

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

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

Scan Summary - NIST SP 800-53

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

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

Scan Summary - OWASP Mobile Top 10 2016

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

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

Scan Summary - Custom

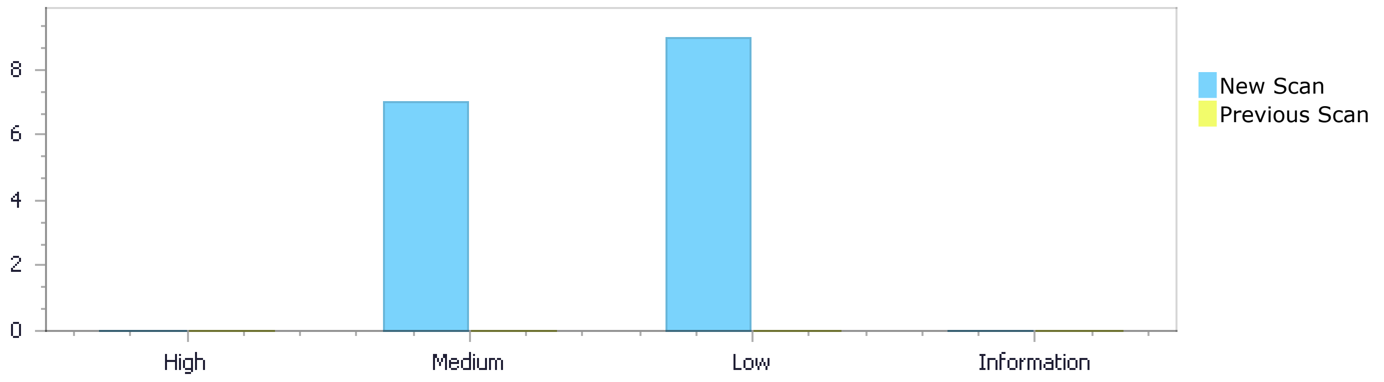
Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

Results Distribution By Status

First scan of the project

	High	Medium	Low	Information	Total
New Issues	0	7	9	0	16
Recurrent Issues	0	0	0	0	0
Total	0	7	9	0	16

Fixed Issues	0	0	0	0	0



Results Distribution By State

	High	Medium	Low	Information	Total
Confirmed	0	0	0	0	0
Not Exploitable	0	0	0	0	0
To Verify	0	7	9	0	16
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	0	7	9	0	16

Result Summary

Vulnerability Type	Occurrences	Severity
Dangerous Functions	5	Medium
Use of Zero Initialized Pointer	2	Medium
NULL Pointer Dereference	9	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
ElectronBot/queue.c	5
ElectronBot/heap_4.c	2

Scan Results Details

Dangerous Functions

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

Categories

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

Description

Dangerous Functions\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=10
Status	New

The dangerous function, memcpy, was found in use at line 2074 in ElectronBot/queue.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2102	2102
Object	memcpy	memcpy

Code Snippet
File Name ElectronBot/queue.c
Method static BaseType_t prvCopyDataToQueue(Queue_t * const pxQueue, const void *pvItemToQueue, const BaseType_t xPosition)

```
....
2102.              ( void ) memcpy( ( void * ) pxQueue->pcWriteTo, 
pvItemToQueue, ( size_t ) pxQueue->uxItemSize ); /*lint !e961 !e418 
!e9087 MISRA exception as the casts are only redundant for some ports, 
plus previous logic ensures a null pointer can only be passed to 
memcpy() if the copy size is 0.  Cast to void required by function 
signature and safe as no alignment requirement and copy length specified 
in bytes. */
```

Dangerous Functions\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=11
Status	New

The dangerous function, memcpy, was found in use at line 2074 in ElectronBot/queue.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2115	2115
Object	memcpy	memcpy

Code Snippet
File Name ElectronBot/queue.c
Method static BaseType_t prvCopyDataToQueue(Queue_t * const pxQueue, const void *pvItemToQueue, const BaseType_t xPosition)

```
....
2115.              ( void ) memcpy( ( void * ) pxQueue-
>u.xQueue.pcReadFrom, pvItemToQueue, ( size_t ) pxQueue->uxItemSize );
/*lint !e961 !e9087 !e418 MISRA exception as the casts are only
redundant for some ports.  Cast to void required by function signature
and safe as no alignment requirement and copy length specified in bytes.
Assert checks null pointer only used when length is 0. */
```

Dangerous Functions\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=12
Status	New

The dangerous function, memcpy, was found in use at line 2153 in ElectronBot/queue.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2166	2166
Object	memcpy	memcpy

Code Snippet
File Name ElectronBot/queue.c
Method static void prvCopyDataFromQueue(Queue_t * const pxQueue, void * const pvBuffer)

```
....
2166.              ( void ) memcpy( ( void * ) pvBuffer, ( void * )
pxQueue->u.xQueue.pcReadFrom, ( size_t ) pxQueue->uxItemSize ); /*lint
!e961 !e418 !e9087 MISRA exception as the casts are only redundant for
some ports.  Also previous logic ensures a null pointer can only be
passed to memcpy() when the count is 0.  Cast to void required by
function signature and safe as no alignment requirement and copy length
specified in bytes. */
```

Dangerous Functions\Path 4:

Severity	Medium
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=13
Status	New

The dangerous function, memcpy, was found in use at line 2450 in ElectronBot/queue.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2500	2500
Object	memcpy	memcpy

Code Snippet

File Name	ElectronBot/queue.c
Method	BaseType_t xQueueCRReceive(QueueHandle_t xQueue, void *pvBuffer, TickType_t xTicksToWait)

```
....
2500.                            ( void ) memcpy( ( void * ) pvBuffer, (
void * ) pxQueue->u.xQueue.pcReadFrom, ( unsigned ) pxQueue->uxItemSize
);
```

Dangerous Functions\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=14
Status	New

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

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2608	2608
Object	memcpy	memcpy

Code Snippet

File Name	ElectronBot/queue.c
Method	BaseType_t xQueueCRReceiveFromISR(QueueHandle_t xQueue, void *pvBuffer, BaseType_t *pxCoRoutineWoken)

```
....
2608.                      ( void ) memcpy( ( void * ) pvBuffer, ( void * )
pxQueue->u.xQueue.pcReadFrom, ( unsigned ) pxQueue->uxItemSize );
```

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=1050075&projectid=50065&pathid=15
Status	New

The variable declared in pvReturn at ElectronBot/heap_4.c in line 115 is not initialized when it is used by pvReturn at ElectronBot/heap_4.c in line 115.

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

Code Snippet

File Name	ElectronBot/heap_4.c
Method	void *pvPortMalloc(size_t xWantedSize)

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

Use of Zero Initialized Pointer\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=16
Status	New

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

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

Code Snippet

File Name	ElectronBot/heap_4.c

Method static void prvHeapInit(void)

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

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=1050075&projectid=50065&pathid=1
Status	New

The variable declared in null at ElectronBot/queue.c in line 2856 is not initialized when it is used by u at ElectronBot/queue.c in line 2153.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2858	2166
Object	null	u

Code Snippet
File Name ElectronBot/queue.c
Method QueueSetMemberHandle_t xQueueSelectFromSet(QueueSetHandle_t xQueueSet, TickType_t const xTicksToWait)

```
....
2858.       QueueSetMemberHandle_t xReturn = NULL;
```

File Name ElectronBot/queue.c
Method static void prvCopyDataFromQueue(Queue_t * const pxQueue, void * const pvBuffer)

```
....
2166.             ( void ) memcpy( ( void * ) pvBuffer, ( void * )
pxQueue->u.xQueue.pcReadFrom, ( size_t ) pxQueue->uxItemSize ); /*lint
!e961 !e418 !e9087 MISRA exception as the casts are only redundant for
some ports.  Also previous logic ensures a null pointer can only be
passed to memcpy() when the count is 0.  Cast to void required by
function signature and safe as no alignment requirement and copy length
specified in bytes. */
```

NULL Pointer Dereference\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=2
Status	New

The variable declared in null at ElectronBot/queue.c in line 2869 is not initialized when it is used by u at ElectronBot/queue.c in line 2153.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2871	2166
Object	null	u

Code Snippet

File Name ElectronBot/queue.c

Method QueueSetMemberHandle_t xQueueSelectFromSetFromISR(QueueSetHandle_t xQueueSet)

```
....
2871.       QueueSetMemberHandle_t xReturn = NULL;
```

File Name ElectronBot/queue.c

Method static void prvCopyDataFromQueue(Queue_t * const pxQueue, void * const pvBuffer)

```
....
2166.             ( void ) memcpy( ( void * ) pvBuffer, ( void * )
pxQueue->u.xQueue.pcReadFrom, ( size_t ) pxQueue->uxItemSize ); /*lint
!e961 !e418 !e9087 MISRA exception as the casts are only redundant for
some ports.  Also previous logic ensures a null pointer can only be
passed to memcpy() when the count is 0.  Cast to void required by
function signature and safe as no alignment requirement and copy length
specified in bytes. */
```

NULL Pointer Dereference\Path 3:

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

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=3
Status	New

The variable declared in null at ElectronBot/queue.c in line 2856 is not initialized when it is used by u at ElectronBot/queue.c in line 2153.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2858	2158
Object	null	u

Code Snippet

File Name ElectronBot/queue.c

Method QueueSetMemberHandle_t xQueueSelectFromSet(QueueSetHandle_t xQueueSet, TickType_t const xTicksToWait)

```
....
2858.       QueueSetMemberHandle_t xReturn = NULL;
```

File Name ElectronBot/queue.c

Method static void prvCopyDataFromQueue(Queue_t * const pxQueue, void * const pvBuffer)

```
....
2158.           if( pxQueue->u.xQueue.pcReadFrom >= pxQueue->u.xQueue.pcTail ) /*lint !e946 MISRA exception justified as use of the relational operator is the cleanest solutions. */
```

NULL Pointer Dereference\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=4
Status	New

The variable declared in null at ElectronBot/queue.c in line 2869 is not initialized when it is used by u at ElectronBot/queue.c in line 2153.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2871	2158
Object	null	u

Code Snippet

File Name ElectronBot/queue.c

Method QueueSetMemberHandle_t xQueueSelectFromSetFromISR(QueueSetHandle_t xQueueSet)

```
....
2871.       QueueSetMemberHandle_t xReturn = NULL;
```

▼

File Name ElectronBot/queue.c

Method static void prvCopyDataFromQueue(Queue_t * const pxQueue, void * const pvBuffer)

```
....
2158.            if( pxQueue->u.xQueue.pcReadFrom >= pxQueue-
>u.xQueue.pcTail ) /*lint !e946 MISRA exception justified as use of the
relational operator is the cleanest solutions. */
```

NULL Pointer Dereference\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=5
Status	New

The variable declared in null at ElectronBot/queue.c in line 2856 is not initialized when it is used by u at ElectronBot/queue.c in line 2153.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2858	2158
Object	null	u

Code Snippet

File Name ElectronBot/queue.c

Method QueueSetMemberHandle_t xQueueSelectFromSet(QueueSetHandle_t xQueueSet, TickType_t const xTicksToWait)

```
....
2858.       QueueSetMemberHandle_t xReturn = NULL;
```

▼

File Name ElectronBot/queue.c

Method static void prvCopyDataFromQueue(Queue_t * const pxQueue, void * const pvBuffer)

```
....
2158.            if( pxQueue->u.xQueue.pcReadFrom >= pxQueue-
>u.xQueue.pcTail ) /*lint !e946 MISRA exception justified as use of the
relational operator is the cleanest solutions. */
```

NULL Pointer Dereference\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=6
Status	New

The variable declared in null at ElectronBot/queue.c in line 2869 is not initialized when it is used by u at ElectronBot/queue.c in line 2153.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2871	2158
Object	null	u

Code Snippet

File Name ElectronBot/queue.c

Method QueueSetMemberHandle_t xQueueSelectFromSetFromISR(QueueSetHandle_t xQueueSet)

```
....
2871.        QueueSetMemberHandle_t xReturn = NULL;
```

File Name ElectronBot/queue.c

Method static void prvCopyDataFromQueue(Queue_t * const pxQueue, void * const pvBuffer)

```
....
2158.            if( pxQueue->u.xQueue.pcReadFrom >= pxQueue->u.xQueue.pcTail ) /*lint !e946 MISRA exception justified as use of the relational operator is the cleanest solutions. */
```

NULL Pointer Dereference\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=7
Status	New

The variable declared in null at ElectronBot/queue.c in line 2856 is not initialized when it is used by pxQueue at ElectronBot/queue.c in line 2291.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	2858	2297

Object	null	pxQueue

Code Snippet
File Name ElectronBot/queue.c
Method QueueSetMemberHandle_t xQueueSelectFromSet(QueueSetHandle_t xQueueSet, TickType_t const xTicksToWait)

```
....
2858.        QueueSetMemberHandle_t xReturn = NULL;
```

File Name ElectronBot/queue.c

Method static BaseType_t prvIsQueueEmpty(const Queue_t *pxQueue)

```
....
2297.             if( pxQueue->uxMessagesWaiting == ( UBaseType_t )  0 )
```

NULL Pointer Dereference\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=8
Status	New

The variable declared in 0 at ElectronBot/heap_4.c in line 333 is not initialized when it is used by xStart at ElectronBot/heap_4.c in line 333.

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

Code Snippet
File Name ElectronBot/heap_4.c
Method static void prvHeapInit(void)

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

NULL Pointer Dereference\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050075&projectid=50065&pathid=9
Status	New

The variable declared in 0 at ElectronBot/queue.c in line 255 is not initialized when it is used by pxQueue at ElectronBot/queue.c in line 255.

	Source	Destination
File	ElectronBot/queue.c	ElectronBot/queue.c
Line	264	264
Object	0	pxQueue

Code Snippet
File Name ElectronBot/queue.c
Method BaseType_t xQueueGenericReset(QueueHandle_t xQueue, BaseType_t xNewQueue)

```
....
264.            pxQueue->uxMessagesWaiting = ( UBaseType_t ) 0U;
```

Dangerous Functions

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

Source Code Examples

CPP

Buffer Overflow in gets()

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

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

Safe reading from user

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

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

Unsafe function for string copy

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

        return 0;
}
```

Safe string copy

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

```
      return 0;
}
```

Unsafe format string

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

Safe format string

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

      return 0;
}
```

Use of Zero Initialized Pointer

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

Source Code Examples

CPP

Explicit NULL Dereference

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

Implicit NULL Dereference

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

Java

Explicit Null Dereference

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

NULL Pointer Dereference

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

Source Code Examples

Scanned Languages

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