# micropython Scan Report

| | |
|---|---|
| Project Name | micropython |
| Scan Start | Friday, June 21, 2024 11:09:19 PM |
| Preset | Checkmarx Default |
| Scan Time | 00h:01m:22s |
| Lines Of Code Scanned | 6730 |
| Files Scanned | 9 |
| Report Creation Time | Friday, June 21, 2024 11:11:28 PM |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056 |
| 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 |

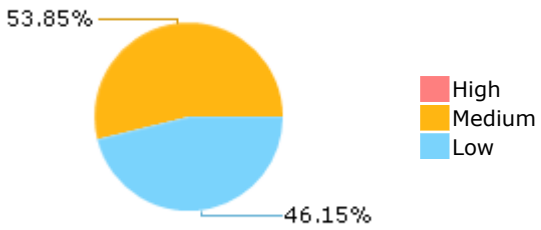
**Results Limit**

Results limit per query was set to 50

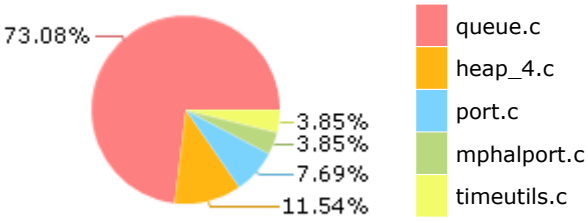
**Selected Queries**

Selected queries are listed in Result Summary

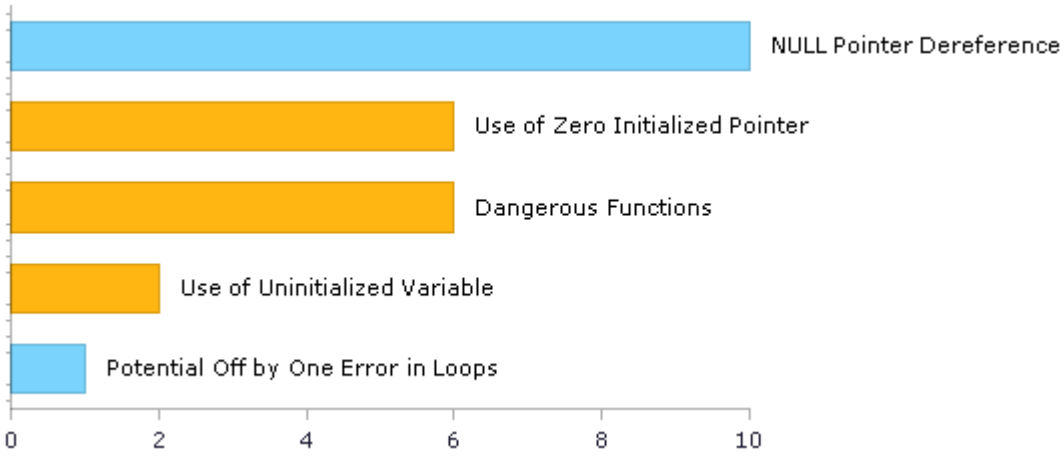
## Result Summary




## 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 | 11 | 5 |
| 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 | 6 | 6 |
| 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 | 6 | 6 |
| 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 | 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)* | 18 | 8 |
| 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) | 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:<br>-Failing to identify the user at all when that should be required<br>-Failure to maintain the user's identity when it is required<br>-Weaknesses in session management | 0 | 0 |
| M5-Insufficient Cryptography | The code applies cryptography to a sensitive information asset. However, the cryptography is insufficient in some way. Note that anything and everything related to TLS or SSL goes in M3. Also, if the app fails to use cryptography at all when it should, that probably belongs in M2. This category is for issues where cryptography was attempted, but it wasnt done correctly. | 0 | 0 |
| M6-Insecure Authorization | This is a category to capture any failures in authorization (e.g., authorization decisions in the client side, forced browsing, etc.). It is distinct from authentication issues (e.g., device enrolment, user identification, etc.).<br>If the app does not authenticate users at all in a situation where it should (e.g., granting anonymous access to some resource or service when authenticated and authorized access is required), then that is an authentication failure not an authorization failure. | 0 | 0 |
| M7-Client Code Quality | This category is the catch-all for code-level implementation problems in the mobile client. That's distinct from server-side coding mistakes. This would capture things like buffer overflows, format string vulnerabilities, and various other code-level mistakes where the solution is to rewrite some code that's running on the mobile device. | 0 | 0 |
| M8-Code Tampering | This category covers binary patching, local resource modification, method hooking, method swizzling, and dynamic memory modification. Once the application is delivered to the mobile device, the code and data resources are resident there. An attacker can either directly modify the code, change the contents of memory dynamically, change or replace the system APIs that the application uses, or | 0 | 0 |

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

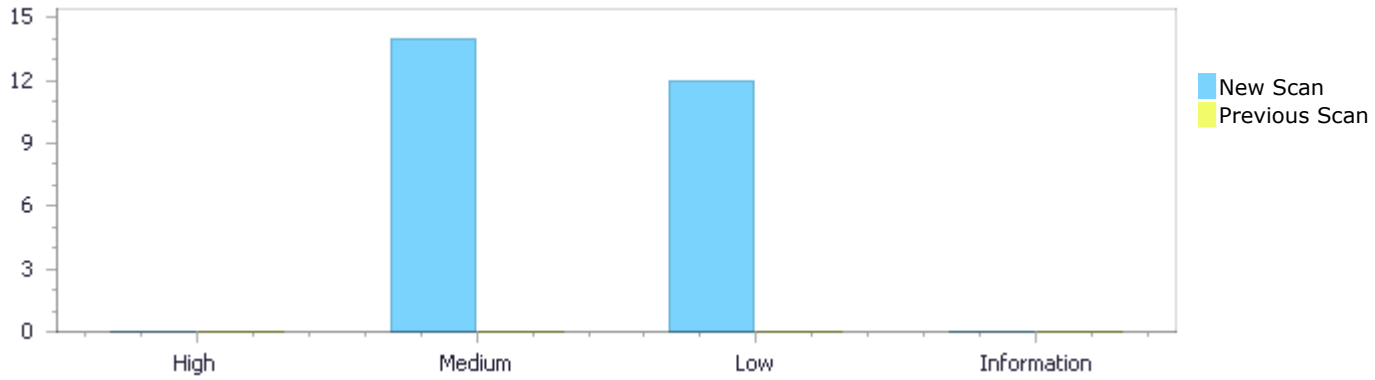
# Scan Summary - Custom

| 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 | 14 | 12 | 0 | 26 |
| Recurrent Issues | 0 | 0 | 0 | 0 | 0 |
| Total | 0 | 14 | 12 | 0 | 26 |
| | | | | | |
| 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 | 14 | 12 | 0 | 26 |
| Urgent | 0 | 0 | 0 | 0 | 0 |
| Proposed Not Exploitable | 0 | 0 | 0 | 0 | 0 |
| Total | 0 | 14 | 12 | 0 | 26 |

## Result Summary

| Vulnerability Type | Occurrences | Severity |
|---|---|---|
| Dangerous Functions | 6 | Medium |
| Use of Zero Initialized Pointer | 6 | Medium |
| Use of Uninitialized Variable | 2 | Medium |
| NULL Pointer Dereference | 10 | Low |
| Potential Off by One Error in Loops | 1 | Low |

| | | |
|---|---|---|
| Use of Sizeof On a Pointer Type | 1 | Low |

# 10 Most Vulnerable Files

## High and Medium Vulnerabilities

| File Name | Issues Found |
|---|---|
| micropython/queue.c | 9 |
| micropython/port.c | 2 |
| micropython/heap_4.c | 2 |
| micropython/mphalport.c | 1 |

# 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=1050066&projectid=50056&pathid=13 |
| Status | New |

The dangerous function, memcpy, was found in use at line 1697 in micropython/queue.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 1725 | 1725 |
| Object | memcpy | memcpy |

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

```
....
1725.             ( void ) memcpy( ( void * ) pxQueue->pcWriteTo,
pvItemToQueue, ( size_t ) pxQueue->uxItemSize ); /*lint !e961 !e418
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. */
```

**Dangerous Functions\Path 2:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=14 |
| Status | New |

The dangerous function, memcpy, was found in use at line 1697 in micropython/queue.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 1738 | 1738 |
| Object | memcpy | memcpy |

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

```
....
1738.             ( void ) memcpy( ( void * ) pxQueue->u.pcReadFrom,
pvItemToQueue, ( size_t ) pxQueue->uxItemSize ); /*lint !e961 MISRA
exception as the casts are only redundant for some ports. */
```

**Dangerous Functions\Path 3:**

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=15
Status New

The dangerous function, memcpy, was found in use at line 1776 in micropython/queue.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 1789 | 1789 |
| Object | memcpy | memcpy |

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

```
....
1789.             ( void ) memcpy( ( void * ) pvBuffer, ( void * )
pxQueue->u.pcReadFrom, ( size_t ) pxQueue->uxItemSize ); /*lint !e961
!e418 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. */
```

**Dangerous Functions\Path 4:**

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=16
Status New

The dangerous function, memcpy, was found in use at line 2071 in micropython/queue.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 2121 | 2121 |
| Object | memcpy | memcpy |

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

```
....
2121.                           ( void ) memcpy( ( void * ) pvBuffer, ( void * ) pxQueue->u.pcReadFrom, ( unsigned ) pxQueue->uxItemSize );
```

**Dangerous Functions\Path 5:**

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=17
Status New

The dangerous function, memcpy, was found in use at line 2209 in micropython/queue.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 2229 | 2229 |
| Object | memcpy | memcpy |

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

```
....
2229.                     ( void ) memcpy( ( void * ) pvBuffer, ( void * ) pxQueue->u.pcReadFrom, ( unsigned ) pxQueue->uxItemSize );
```

**Dangerous Functions\Path 6:**

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=18

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

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

| | Source | Destination |
|---|---|---|
| File | micropython/mphalport.c | micropython/mphalport.c |
| Line | 213 | 213 |
| Object | strlen | strlen |

Code Snippet
File Name micropython/mphalport.c
Method void mp_hal_stdout_tx_str(const char *str) {

```
....
213.      mp_hal_stdout_tx_strn(str, strlen(str));
```

## 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=1050066&projectid=50056&pathid=21 |
| Status | New |

The variable declared in pvReturn at micropython/heap_4.c in line 155 is not initialized when it is used by pvReturn at micropython/heap_4.c in line 155.

| | Source | Destination |
|---|---|---|
| File | micropython/heap_4.c | micropython/heap_4.c |
| Line | 158 | 300 |
| Object | pvReturn | pvReturn |

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

```
....
158.  void *pvReturn = NULL;
....
300.        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=1050066&projectid=50056&pathid=22 |
| Status | New |

The variable declared in xReturn at micropython/queue.c in line 2477 is not initialized when it is used by pcTail at micropython/queue.c in line 1697.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 2479 | 1713 |
| Object | xReturn | pcTail |

Code Snippet

File Name micropython/queue.c

Method QueueSetMemberHandle_t xQueueSelectFromSet( QueueSetHandle_t xQueueSet, TickType_t const xTicksToWait )

```
....
2479.       QueueSetMemberHandle_t xReturn = NULL;
```

File Name micropython/queue.c

Method static BaseType_t prvCopyDataToQueue( Queue_t * const pxQueue, const void *pvItemToQueue, const BaseType_t xPosition )

```
....
1713.                       xReturn = xTaskPriorityDisinherit( ( void
* ) pxQueue->pxMutexHolder );
```

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

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=23 |
| Status | New |

The variable declared in pcTail at micropython/queue.c in line 1697 is not initialized when it is used by pcTail at micropython/queue.c in line 1697.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 1714 | 1713 |
| Object | pcTail | pcTail |

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

```
....
1714.                           pxQueue->pxMutexHolder = NULL;
....
1713.                           xReturn = xTaskPriorityDisinherit( ( void
* ) pxQueue->pxMutexHolder );
```

**Use of Zero Initialized Pointer\Path 4:**

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=24
Status New

The variable declared in xReturn at micropython/queue.c in line 2477 is not initialized when it is used by pcTail at micropython/queue.c in line 1697.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 2479 | 1742 |
| Object | xReturn | pcTail |

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

```
....
2479.       QueueSetMemberHandle_t xReturn = NULL;
```

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

```
....
1742.                   pxQueue->u.pcReadFrom = ( pxQueue->pcTail -
pxQueue->uxItemSize );
```

**Use of Zero Initialized Pointer\Path 5:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=25 |
| Status | New |

The variable declared in pcTail at micropython/queue.c in line 1697 is not initialized when it is used by pcTail at micropython/queue.c in line 1697.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 1714 | 1742 |
| Object | pcTail | pcTail |

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

```
....
1714.                           pxQueue->pxMutexHolder = NULL;
....
1742.                    pxQueue->u.pcReadFrom = ( pxQueue->pcTail -
pxQueue->uxItemSize );
```

**Use of Zero Initialized Pointer\Path 6:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=26 |
| Status | New |

The variable declared in pxNextFreeBlock at micropython/heap_4.c in line 371 is not initialized when it is used by pxNextFreeBlock at micropython/heap_4.c in line 371.

| | Source | Destination |
|---|---|---|
| File | micropython/heap_4.c | micropython/heap_4.c |
| Line | 402 | 408 |
| Object | pxNextFreeBlock | pxNextFreeBlock |

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

```
....
402.        pxEnd->pxNextFreeBlock = NULL;
....
408.        pxFirstFreeBlock->pxNextFreeBlock = pxEnd;
```

## Use of Uninitialized Variable

Query Path:
CPP\Cx\CPP Medium Threat\Use of Uninitialized Variable Version:0

### Categories

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

*Description*

**Use of Uninitialized Variable\Path 1:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=19 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | micropython/port.c | micropython/port.c |
| Line | 635 | 645 |
| Object | ulCurrentInterrupt | ulCurrentInterrupt |

Code Snippet
File Name micropython/port.c
Method void vPortValidateInterruptPriority( void )

```
....
635.        uint32_t ulCurrentInterrupt;
....
645.                    ucCurrentPriority =
pcInterruptPriorityRegisters[ ulCurrentInterrupt ];
```

**Use of Uninitialized Variable\Path 2:**

| | |
|---|---|
| Severity | Medium |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=20 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | micropython/port.c | micropython/port.c |
| Line | 635 | 642 |
| Object | ulCurrentInterrupt | ulCurrentInterrupt |

| | |
|---|---|
| Code Snippet | |
| File Name | micropython/port.c |
| Method | void vPortValidateInterruptPriority( void ) |

```
....
635.        uint32_t ulCurrentInterrupt;
....
642.              if( ulCurrentInterrupt >=
portFIRST_USER_INTERRUPT_NUMBER )
```

## 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=1050066&projectid=50056&pathid=3 |
| Status | New |

The variable declared in null at micropython/queue.c in line 627 is not initialized when it is used by u at micropython/queue.c in line 1776.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 646 | 1789 |
| Object | null | u |

| | |
|---|---|
| Code Snippet | |
| File Name | micropython/queue.c |
| Method | BaseType_t xQueueTakeMutexRecursive( QueueHandle_t xMutex, TickType_t xTicksToWait ) |

```
....
646.                    xReturn = xQueueGenericReceive( pxMutex, NULL,
xTicksToWait, pdFALSE );
```

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

```
....
1789.             ( void ) memcpy( ( void * ) pvBuffer, ( void * )
pxQueue->u.pcReadFrom, ( size_t ) pxQueue->uxItemSize ); /*lint !e961
!e418 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. */
```

**NULL Pointer Dereference\Path 2:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=4 |
| Status | New |

The variable declared in null at micropython/queue.c in line 2477 is not initialized when it is used by u at micropython/queue.c in line 1776.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 2479 | 1789 |
| Object | null | u |

Code Snippet

File Name micropython/queue.c

Method QueueSetMemberHandle_t xQueueSelectFromSet( QueueSetHandle_t xQueueSet, TickType_t const xTicksToWait )

```
....
2479.       QueueSetMemberHandle_t xReturn = NULL;
```

File Name micropython/queue.c

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

```
....
1789.             ( void ) memcpy( ( void * ) pvBuffer, ( void * )
pxQueue->u.pcReadFrom, ( size_t ) pxQueue->uxItemSize ); /*lint !e961
!e418 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. */
```

**NULL Pointer Dereference\Path 3:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=5 |
| Status | New |

The variable declared in null at micropython/queue.c in line 2490 is not initialized when it is used by u at micropython/queue.c in line 1776.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 2492 | 1789 |
| Object | null | u |

Code Snippet
File Name micropython/queue.c
Method QueueSetMemberHandle_t xQueueSelectFromSetFromISR( QueueSetHandle_t xQueueSet )

```
....
2492.       QueueSetMemberHandle_t xReturn = NULL;
```

File Name micropython/queue.c

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

```
....
1789.             ( void ) memcpy( ( void * ) pvBuffer, ( void * )
pxQueue->u.pcReadFrom, ( size_t ) pxQueue->uxItemSize ); /*lint !e961
!e418 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. */
```

**NULL Pointer Dereference\Path 4:**

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=6
Status New

The variable declared in null at micropython/queue.c in line 627 is not initialized when it is used by u at micropython/queue.c in line 1776.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 646 | 1781 |
| Object | null | u |

Code Snippet
File Name micropython/queue.c
Method BaseType_t xQueueTakeMutexRecursive( QueueHandle_t xMutex, TickType_t xTicksToWait )

```
....
646.                    xReturn = xQueueGenericReceive( pxMutex, NULL,
xTicksToWait, pdFALSE );
```

File Name micropython/queue.c

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

```
....
1781.             if( pxQueue->u.pcReadFrom >= pxQueue->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=1050066&projectid=50056&pathid=7 |
| Status | New |

The variable declared in null at micropython/queue.c in line 2477 is not initialized when it is used by u at micropython/queue.c in line 1776.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 2479 | 1781 |
| Object | null | u |

Code Snippet

File Name micropython/queue.c

Method QueueSetMemberHandle_t xQueueSelectFromSet( QueueSetHandle_t xQueueSet, TickType_t const xTicksToWait )

```
....
2479.       QueueSetMemberHandle_t xReturn = NULL;
```

File Name micropython/queue.c

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

```
....
1781.             if( pxQueue->u.pcReadFrom >= pxQueue->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=1050066&projectid=50056&pathid=8 |
| Status | New |

The variable declared in null at micropython/queue.c in line 2490 is not initialized when it is used by u at micropython/queue.c in line 1776.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 2492 | 1781 |
| Object | null | u |

Code Snippet

File Name micropython/queue.c

Method QueueSetMemberHandle_t xQueueSelectFromSetFromISR( QueueSetHandle_t xQueueSet )

```
....
2492.       QueueSetMemberHandle_t xReturn = NULL;
```

File Name micropython/queue.c

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

```
....
1781.             if( pxQueue->u.pcReadFrom >= pxQueue->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=1050066&projectid=50056&pathid=9 |
| Status | New |

The variable declared in null at micropython/queue.c in line 627 is not initialized when it is used by pxQueue at micropython/queue.c in line 1914.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 646 | 1920 |
| Object | null | pxQueue |

Code Snippet
File Name micropython/queue.c
Method BaseType_t xQueueTakeMutexRecursive( QueueHandle_t xMutex, TickType_t xTicksToWait )

```
....
646.                     xReturn = xQueueGenericReceive( pxMutex, NULL, xTicksToWait, pdFALSE );
```

File Name micropython/queue.c

Method static BaseType_t prvIsQueueEmpty( const Queue_t *pxQueue )

```
....
1920.             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=1050066&projectid=50056&pathid=10 |
| Status | New |

The variable declared in null at micropython/queue.c in line 2477 is not initialized when it is used by pxQueue at micropython/queue.c in line 1914.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 2479 | 1920 |
| Object | null | pxQueue |

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

```
....
2479.       QueueSetMemberHandle_t xReturn = NULL;
```

File Name micropython/queue.c

Method static BaseType_t prvIsQueueEmpty( const Queue_t *pxQueue )

```
....
1920.             if( pxQueue->uxMessagesWaiting == ( UBaseType_t )  0 )
```

**NULL Pointer Dereference\Path 9:**

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

| | |
|---|---|
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=11 |
| Status | New |

The variable declared in 0 at micropython/heap_4.c in line 371 is not initialized when it is used by xStart at micropython/heap_4.c in line 371.

| | Source | Destination |
|---|---|---|
| File | micropython/heap_4.c | micropython/heap_4.c |
| Line | 393 | 393 |
| Object | 0 | xStart |

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

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

**NULL Pointer Dereference\Path 10:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=12 |
| Status | New |

The variable declared in 0 at micropython/queue.c in line 279 is not initialized when it is used by pxQueue at micropython/queue.c in line 279.

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 288 | 288 |
| Object | 0 | pxQueue |

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

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

## Use of Sizeof On a Pointer Type

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

*Description*

**Use of Sizeof On a Pointer Type\Path 1:**

| | |
|---|---|
| Severity | Low |
| Result State | To Verify |
| Online Results | http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050066&projectid=50056&pathid=1 |
| Status | New |

| | Source | Destination |
|---|---|---|
| File | micropython/queue.c | micropython/queue.c |
| Line | 2398 | 2398 |
| Object | sizeof | sizeof |

Code Snippet
File Name micropython/queue.c
Method QueueSetHandle_t xQueueCreateSet( const UBaseType_t uxEventQueueLength )

```
....
2398.             pxQueue = xQueueGenericCreate( uxEventQueueLength,
sizeof( Queue_t * ), queueQUEUE_TYPE_SET );
```

## 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=1050066&projectid=50056&pathid=2 |
| Status | New |

The buffer allocated by <= in micropython/timeutils.c at line 68 does not correctly account for the actual size of the value, resulting in an incorrect allocation that is off by one.

| | Source | Destination |
|---|---|---|
| File | micropython/timeutils.c | micropython/timeutils.c |
| Line | 131 | 131 |
| Object | <= | <= |

Code Snippet
File Name micropython/timeutils.c

| Method | void timeutils_seconds_since_2000_to_struct_time(mp_uint_t t, timeutils_struct_time_t *tm) { |
|---|---|

```
....
131.      for (month = 0; days_in_month[month] <= days; month++) {
```

# 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 Uninitialized Variable

**Weakness ID:** 457 *(Weakness Variant)* **Status:** Draft

### Description

### Description Summary

The code uses a variable that has not been initialized, leading to unpredictable or unintended results.

### Extended Description

In some languages, such as C, an uninitialized variable contains contents of previously-used memory. An attacker can sometimes control or read these contents.

### Time of Introduction

- Implementation

### Applicable Platforms

### Languages

C: *(Sometimes)*

C++: *(Sometimes)*

Perl: *(Often)*

All

### Common Consequences

| Scope | Effect |
|---|---|
| Availability<br>Integrity | Initial variables usually contain junk, which can not be trusted for consistency. This can lead to denial of service conditions, or modify control flow in unexpected ways. In some cases, an attacker can "pre-initialize" the variable using previous actions, which might enable code execution. This can cause a race condition if a lock variable check passes when it should not. |
| Authorization | Strings that are not initialized are especially dangerous, since many functions expect a null at the end -- and only at the end -- of a string. |

### Likelihood of Exploit

High

### Demonstrative Examples

### Example 1

The following switch statement is intended to set the values of the variables aN and bN, but in the default case, the programmer has accidentally set the value of aN twice. As a result, bN will have an undefined value.

*(Bad Code)*

*Example Language:* **C**

```
switch (ctl) {
case -1:
aN = 0;
bN = 0;
break;
case 0:
aN = i;
bN = -i;
break;
case 1:
aN = i + NEXT_SZ;
bN = i - NEXT_SZ;
break;
default:
```

```
aN = -1;
aN = -1;
break;
}
repaint(aN, bN);
```

Most uninitialized variable issues result in general software reliability problems, but if attackers can intentionally trigger the use of an uninitialized variable, they might be able to launch a denial of service attack by crashing the program. Under the right circumstances, an attacker may be able to control the value of an uninitialized variable by affecting the values on the stack prior to the invocation of the function.

## Example 2

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

```
int foo;
void bar() {
if (foo==0)
/.../
/../
}
```

### Observed Examples

| Reference | Description |
|---|---|
| CVE-2008-0081 | Uninitialized variable leads to code execution in popular desktop application. |
| CVE-2007-4682 | Crafted input triggers dereference of an uninitialized object pointer. |
| CVE-2007-3468 | Crafted audio file triggers crash when an uninitialized variable is used. |
| CVE-2007-2728 | Uninitialized random seed variable used. |

### Potential Mitigations

**Phase: Implementation**

Assign all variables to an initial value.

**Phase: Build and Compilation**

Most compilers will complain about the use of uninitialized variables if warnings are turned on.

**Phase: Requirements**

The choice could be made to use a language that is not susceptible to these issues.

**Phase: Architecture and Design**

Mitigating technologies such as safe string libraries and container abstractions could be introduced.

### Other Notes

Before variables are initialized, they generally contain junk data of what was left in the memory that the variable takes up. This data is very rarely useful, and it is generally advised to pre-initialize variables or set them to their first values early. If one forgets -- in the C language -- to initialize, for example a char *, many of the simple string libraries may often return incorrect results as they expect the null termination to be at the end of a string.

Stack variables in C and C++ are not initialized by default. Their initial values are determined by whatever happens to be in their location on the stack at the time the function is invoked. Programs should never use the value of an uninitialized variable. It is not uncommon for programmers to use an uninitialized variable in code that handles errors or other rare and exceptional circumstances. Uninitialized variable warnings can sometimes indicate the presence of a typographic error in the code.

### 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 | Weakness Base | 456 | Missing Initialization | **Development Concepts (primary)699 Research Concepts** |

| | | | | |
|---|---|---|---|---|
| | | | | **(primary)1000** |
| MemberOf | View | 630 | Weaknesses Examined by SAMATE | **Weaknesses Examined by SAMATE (primary)630** |

## Taxonomy Mappings

| Mapped Taxonomy Name | Node ID | Fit | Mapped Node Name |
|---|---|---|---|
| CLASP | | | Uninitialized variable |
| 7 Pernicious Kingdoms | | | Uninitialized Variable |

## White Box Definitions

A weakness where the code path has:

1. start statement that defines variable
2. end statement that accesses the variable
3. the code path does not contain a statement that assigns value to the variable

## References


mercy. "Exploiting Uninitialized Data". Jan 2006. < http://www.felinemenace.org/~mercy/papers/UBehavior/UBehavior.zip>.

Microsoft Security Vulnerability Research & Defense. "MS08-014 : The Case of the Uninitialized Stack Variable Vulnerability". 2008-03-11. <http://blogs.technet.com/swi/archive/2008/03/11/the-case-of-the-uninitialized-stack-variable-vulnerability.aspx>.


## 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, Description, Relationships, Observed Example, Other Notes, References, Taxonomy Mappings | | |
| 2009-01-12 | CWE Content Team | MITRE | Internal |
| | updated Common Consequences, Demonstrative Examples, Potential Mitigations | | |
| 2009-03-10 | CWE Content Team | MITRE | Internal |
| | updated Demonstrative Examples | | |
| 2009-05-27 | CWE Content Team | MITRE | Internal |
| | updated Demonstrative Examples | | |

**Previous Entry Names**

| Change Date | Previous Entry Name |
|---|---|
| 2008-04-11 | Uninitialized Variable |

BACK TO TOP

# 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 sizeof() on a Pointer Type

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

### Description

### Description Summary

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

### Time of Introduction

- Implementation

### Applicable Platforms

### Languages

C

C++

### Common Consequences

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

### Likelihood of Exploit

High

### Demonstrative Examples

### Example 1

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

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

*(Bad Code)*

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

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

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

*(Good Code)*

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

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

### Example 2

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

*(Bad Code)*

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

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

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

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

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

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

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

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

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

*(Attack)*

```
pass5
passABCDEFGH
passWORD
```

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

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

## Potential Mitigations

**Phase: Implementation**

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

## Other Notes

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

## Weakness Ordinalities

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

## Relationships

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

## Taxonomy Mappings

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

## White Box Definitions

A weakness where code path has:

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

## References

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

## Content History

**Submissions**

| Submission Date | Submitter | Organization | Source |
|---|---|---|---|
| | CLASP | | Externally Mined |

**Modifications**

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

BACK TO TOP

# Potential Off by One Error in Loops

## Risk

### What might happen

An off by one error may result in overwriting or over-reading of unintended memory; in most cases, this can result in unexpected behavior and even application crashes. In other cases, where allocation can be controlled by an attacker, a combination of variable assignment and an off by one error can result in execution of malicious code.

---

## Cause

### How does it happen

Often when designating variables to memory, a calculation error may occur when determining size or length that is off by one.

For example in loops, when allocating an array of size 2, its cells are counted as 0,1 - therefore, if a For loop iterator on the array is incorrectly set with the start condition i=0 and the continuation condition i<=2, three cells will be accessed instead of 2, and an attempt will be made to write or read cell [2], which was not originally allocated, resulting in potential corruption of memory outside the bounds of the originally assigned array.

Another example occurs when a null-byte terminated string, in the form of a character array, is copied without its terminating null-byte. Without the null-byte, the string representation is unterminated, resulting in certain functions to over-read memory as they expect the missing null terminator.

---

## General Recommendations

### How to avoid it

- Always ensure that a given iteration boundary is correct:
  - With array iterations, consider that arrays begin with cell 0 and end with cell n-1, for a size n array.
  - With character arrays and null-byte terminated string representations, consider that the null byte is required and should not be overwritten or ignored; ensure functions in use are not vulnerable to off-by-one, specifically for instances where null-bytes are automatically appended after the buffer, instead of in place of its last character.
- Where possible, use safe functions that manage memory and are not prone to off-by-one errors.

---

## Source Code Examples

### CPP

### Off-By-One in For Loop

```
int *ptr;
ptr = (int*)malloc(5 * sizeof(int));
for (int i = 0; i <= 5; i++)
{
      ptr[i] = i * 2 + 1; // ptr[5] will be set, but is out of bounds
```

```
}
```

### Proper Iteration in For Loop

```
int *ptr;
ptr = (int*)malloc(5 * sizeof(int));
for (int i = 0; i < 5; i++)
{
      ptr[i] = i * 2 + 1; // ptr[0-4] are well defined
}
```

### Off-By-One in strncat

```
strncat(buf, input, sizeof(buf) - strlen(buf)); // actual value should be sizeof(buf)-
strlen(buf)-1 - this form will overwrite the terminating nullbyte
```

# NULL Pointer Dereference

## Risk

### What might happen

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

---

## Cause

### How does it happen

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

---

## General Recommendations

### How to avoid it

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

---

## Source Code Examples

## Scanned Languages

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