flipperzero-firmware Scan Report

Project Name	flipperzero-firmware
Scan Start	Friday, June 21, 2024 11:18:56 PM
Preset	Checkmarx Default
Scan Time	00h:01m:26s
Lines Of Code Scanned	6093
Files Scanned	12
Report Creation Time	Friday, June 21, 2024 11:22:00 PM
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062
Team	CxServer
Checkmarx Version	8.7.0
Scan Type	Full
Source Origin	LocalPath
Density	8/1000 (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

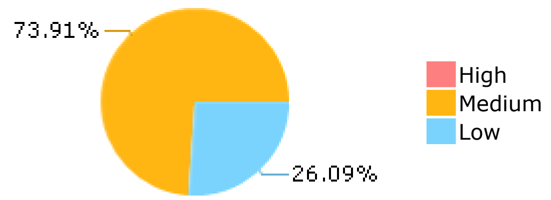
Results Limit

Results limit per query was set to 50

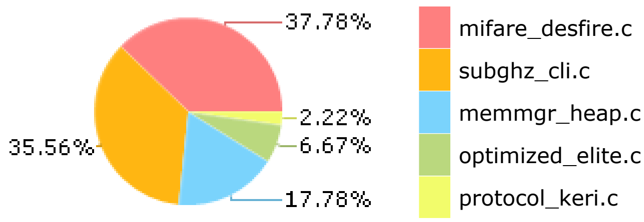
Selected Queries

Selected queries are listed in Result Summary

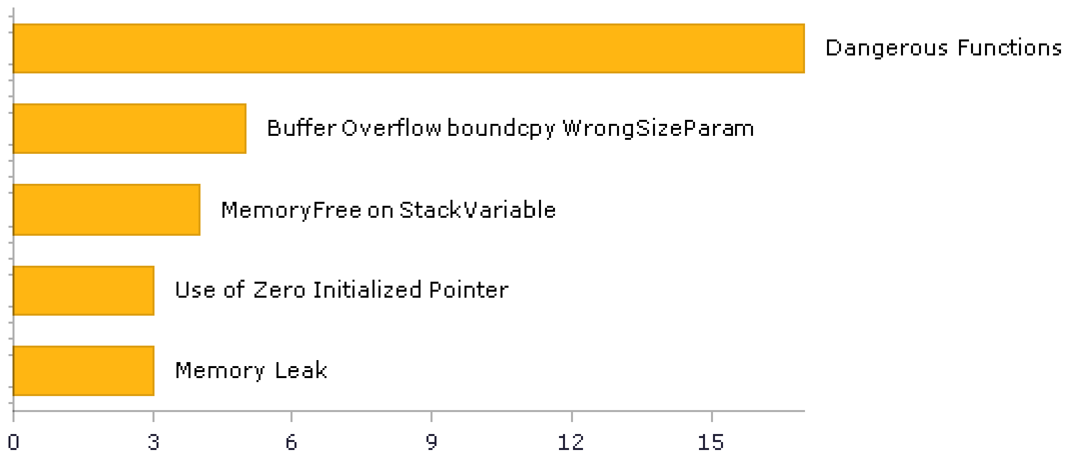
Result Summary



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	6	6
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	17	17
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	17	17
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	7	7
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)*	7	7
SC-8 Transmission Confidentiality and Integrity (P1)	0	0
SI-10 Information Input Validation (P1)*	3	3
SI-11 Error Handling (P2)*	10	10
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

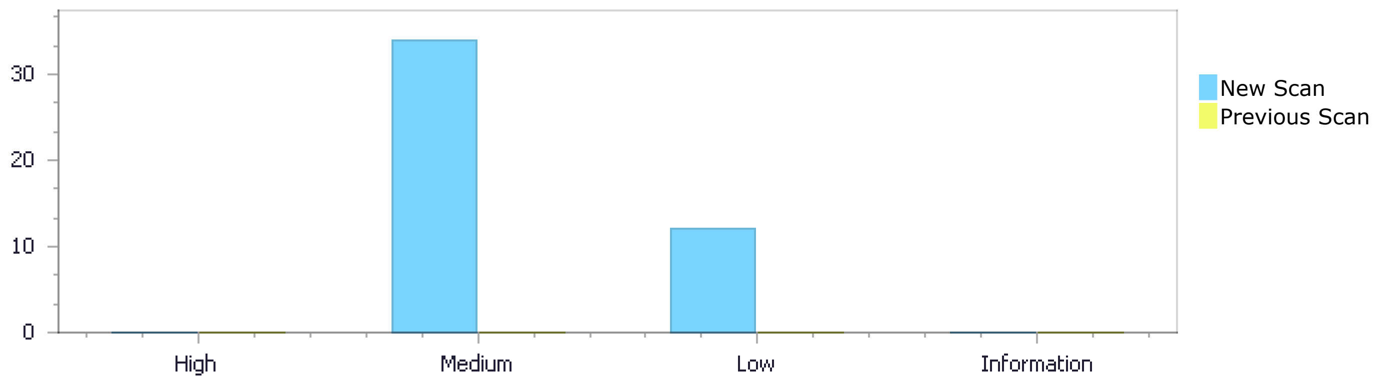
Scan Summary - Custom

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	34	12	0	46
Recurrent Issues	0	0	0	0	0
Total	0	34	12	0	46
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	34	12	0	46
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	0	34	12	0	46

Result Summary

Vulnerability Type	Occurrences	Severity
Dangerous Functions	17	Medium
Buffer Overflow boundcpy WrongSizeParam	5	Medium
MemoryFree on StackVariable	4	Medium
Memory Leak	3	Medium
Use of Zero Initialized Pointer	3	Medium

Char Overflow	2	Medium
Unchecked Return Value	10	Low
NULL Pointer Dereference	1	Low
Unchecked Array Index	1	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
flipperzero-firmware/mifare_desfire.c	14
flipperzero-firmware/subghz_cli.c	11
flipperzero-firmware/memmgr_heap.c	6
flipperzero-firmware/optimized_elite.c	3

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=1050072&projectid=50062&pathid=24
Status	New

The dangerous function, memcpy, was found in use at line 270 in flipperzero-firmware/mifare_desfire.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	279	279
Object	memcpy	memcpy

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method bool mf_df_parse_get_version_response(uint8_t* buf, uint16_t len, MifareDesfireVersion* out) {

```
....
279.        memcpy(out, buf, sizeof(MifareDesfireVersion));
```

Dangerous Functions\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=25
Status	New

The dangerous function, memcpy, was found in use at line 347 in flipperzero-firmware/mifare_desfire.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c

Line	362	362
Object	memcpy	memcpy

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method bool mf_df_parse_get_application_ids_response(

```
....
362.            memcpy(app->id, buf, 3);
```

Dangerous Functions\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=26
Status	New

The dangerous function, memcpy, was found in use at line 485 in flipperzero-firmware/mifare_desfire.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	492	492
Object	memcpy	memcpy

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method bool mf_df_parse_read_data_response(uint8_t* buf, uint16_t len, MifareDesfireFile* out) {

```
....
492.        memcpy(out->contents, buf, len);
```

Dangerous Functions\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=27
Status	New

The dangerous function, memcpy, was found in use at line 188 in flipperzero-firmware/optimized_elite.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/optimized_elite.c	flipperzero-firmware/optimized_elite.c
Line	228	228

Object	memcpy	memcpy

Code Snippet
File Name flipperzero-firmware/optimized_elite.c
Method void loclass_hash2(uint8_t* key64, uint8_t* outp_keytable) {

```
....
228.                memcpy(outp_keytable + i * 16, y[i], 8);
```

Dangerous Functions\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=28
Status	New

The dangerous function, memcpy, was found in use at line 188 in flipperzero-firmware/optimized_elite.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/optimized_elite.c	flipperzero-firmware/optimized_elite.c
Line	229	229
Object	memcpy	memcpy

Code Snippet
File Name flipperzero-firmware/optimized_elite.c
Method void loclass_hash2(uint8_t* key64, uint8_t* outp_keytable) {

```
....
229.                memcpy(outp_keytable + 8 + i * 16, z[i], 8);
```

Dangerous Functions\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=29
Status	New

The dangerous function, memcpy, was found in use at line 156 in flipperzero-firmware/optimized_elite.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/optimized_elite.c	flipperzero-firmware/optimized_elite.c
Line	157	157
Object	memcpy	memcpy

Code Snippet
File Name flipperzero-firmware/optimized_elite.c
Method static void loclass_rk(uint8_t* key, uint8_t n, uint8_t* outp_key) {

```
....
157.        memcpy(outp_key, key, 8);
```

Dangerous Functions\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=30
Status	New

The dangerous function, memcpy, was found in use at line 898 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	943	943
Object	memcpy	memcpy

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method void subghz_on_system_start() {

```
....
943.            memcpy(
```

Dangerous Functions\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=31
Status	New

The dangerous function, sscanf, was found in use at line 27 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	32	32
Object	sscanf	sscanf

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method void subghz_cli_command_tx_carrier(Cli* cli, FuriString* args, void* context) {

```
....
32.             int ret = sscanf(furi_string_get_cstr(args), "%lu",
&frequency);
```

Dangerous Functions\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=32
Status	New

The dangerous function, sscanf, was found in use at line 71 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	76	76
Object	sscanf	sscanf

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method void subghz_cli_command_rx_carrier(Cli* cli, FuriString* args, void* context) {

```
....
76.             int ret = sscanf(furi_string_get_cstr(args), "%lu",
&frequency);
```

Dangerous Functions\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=33
Status	New

The dangerous function, sscanf, was found in use at line 112 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	121	121
Object	sscanf	sscanf

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method void subghz_cli_command_tx(Cli* cli, FuriString* args, void* context) {

```
....
121.                sscanf(furi_string_get_cstr(args), "%lx %lu %lu %lu",
&key, &frequency, &te, &repeat);
```

Dangerous Functions\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=34
Status	New

The dangerous function, sscanf, was found in use at line 233 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	238	238
Object	sscanf	sscanf

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method void subghz_cli_command_rx(Cli* cli, FuriString* args, void* context) {

```
....
238.            int ret = sscanf(furi_string_get_cstr(args), "%lu",
&frequency);
```

Dangerous Functions\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=35
Status	New

The dangerous function, sscanf, was found in use at line 317 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	322	322
Object	sscanf	sscanf

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method void subghz_cli_command_rx_raw(Cli* cli, FuriString* args, void* context) {

```
....
322.            int ret = sscanf(furi_string_get_cstr(args), "%lu",
&frequency);
```

Dangerous Functions\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=36
Status	New

The dangerous function, sscanf, was found in use at line 612 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	616	616
Object	sscanf	sscanf

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method static void subghz_cli_command_chat(Cli* cli, FuriString* args) {

```
....
616.            int ret = sscanf(furi_string_get_cstr(args), "%lu",
&frequency);
```

Dangerous Functions\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=37
Status	New

The dangerous function, strlen, was found in use at line 612 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	712	712
Object	strlen	strlen

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method static void subghz_cli_command_chat(Cli* cli, FuriString* args) {

```
....
712.                    strlen(furi_string_get_cstr(input)))) {
```

Dangerous Functions\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=38
Status	New

The dangerous function, strlen, was found in use at line 612 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	756	756
Object	strlen	strlen

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method static void subghz_cli_command_chat(Cli* cli, FuriString* args) {

```
....
756.                    strlen(furi_string_get_cstr(sysmsg)));
```

Dangerous Functions\Path 16:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=39
Status	New

The dangerous function, strlen, was found in use at line 612 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	764	764
Object	strlen	strlen

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method static void subghz_cli_command_chat(Cli* cli, FuriString* args) {

```
....
764.                  strlen(furi_string_get_cstr(sysmsg)));
```

Dangerous Functions\Path 17:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=40
Status	New

The dangerous function, realloc, was found in use at line 864 in flipperzero-firmware/subghz_cli.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	878	878
Object	realloc	realloc

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method static bool subghz_on_system_start_istream_decode_band(

```
....
878.       region = realloc( //-V701
```

Buffer Overflow boundcpy WrongSizeParam

Query Path:
CPP\Cx\CPP Buffer Overflow\Buffer Overflow boundcpy WrongSizeParam Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
OWASP Top 10 2017: A1-Injection

Description

Buffer Overflow boundcpy WrongSizeParam\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=11
Status	New

The size of the buffer used by mf_df_parse_get_version_response in MifareDesfireVersion, at line 270 of flipperzero-firmware/mifare_desfire.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that mf_df_parse_get_version_response passes to MifareDesfireVersion, at line 270 of flipperzero-firmware/mifare_desfire.c, to overwrite the target buffer.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c

Line	279	279
Object	MifareDesfireVersion	MifareDesfireVersion

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method bool mf_df_parse_get_version_response(uint8_t* buf, uint16_t len, MifareDesfireVersion* out) {

```
....
279.        memcpy(out, buf, sizeof(MifareDesfireVersion));
```

Buffer Overflow boundcpy WrongSizeParam\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=12
Status	New

The size of the buffer used by mf_df_read_card in MifareDesfireKeySettings, at line 496 of flipperzero-firmware/mifare_desfire.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that mf_df_read_card passes to MifareDesfireKeySettings, at line 496 of flipperzero-firmware/mifare_desfire.c, to overwrite the target buffer.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	585	585
Object	MifareDesfireKeySettings	MifareDesfireKeySettings

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method bool mf_df_read_card(FuriHalNfcTxRxContext* tx_rx, MifareDesfireData* data) {

```
....
585.                    memset(app->key_settings, 0,
sizeof(MifareDesfireKeySettings));
```

Buffer Overflow boundcpy WrongSizeParam\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=13
Status	New

The size of the buffer used by mf_df_parse_get_application_ids_response in MifareDesfireApplication, at line 347 of flipperzero-firmware/mifare_desfire.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that mf_df_parse_get_application_ids_response passes to MifareDesfireApplication, at line 347 of flipperzero-firmware/mifare_desfire.c, to overwrite the target buffer.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	361	361
Object	MifareDesfireApplication	MifareDesfireApplication

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method bool mf_df_parse_get_application_ids_response(

```
....
361.            memset(app, 0, sizeof(MifareDesfireApplication));
```

Buffer Overflow boundcpy WrongSizeParam\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=14
Status	New

The size of the buffer used by mf_df_parse_get_file_ids_response in MifareDesfireFile, at line 388 of flipperzero-firmware/mifare_desfire.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that mf_df_parse_get_file_ids_response passes to MifareDesfireFile, at line 388 of flipperzero-firmware/mifare_desfire.c, to overwrite the target buffer.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	396	396
Object	MifareDesfireFile	MifareDesfireFile

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method bool mf_df_parse_get_file_ids_response(uint8_t* buf, uint16_t len, MifareDesfireFile** file_head) {

```
....
396.            memset(file, 0, sizeof(MifareDesfireFile));
```

Buffer Overflow boundcpy WrongSizeParam\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=15
Status	New

The size of the buffer used by pvPortMalloc in to_wipe, at line 339 of flipperzero-firmware/memmgr_heap.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pvPortMalloc passes to to_wipe, at line 339 of flipperzero-firmware/memmgr_heap.c, to overwrite the target buffer.

	Source	Destination
File	flipperzero-firmware/memmgr_heap.c	flipperzero-firmware/memmgr_heap.c
Line	485	485
Object	to_wipe	to_wipe

Code Snippet
File Name flipperzero-firmware/memmgr_heap.c
Method void* pvPortMalloc(size_t xWantedSize) {

```
....
485.       pvReturn = memset(pvReturn, 0, to_wipe);
```

MemoryFree on StackVariable

Query Path:
CPP\Cx\CPP Medium Threat\MemoryFree on StackVariable Version:0
Description

MemoryFree on StackVariable\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=16
Status	New

Calling free() (line 7) on a variable that was not dynamically allocated (line 7) in file flipperzero-firmware/mifare_desfire.c may result with a crash.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	13	13
Object	key_version	key_version

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method void mf_df_clear(MifareDesfireData* data) {

```
....
13.              free(key_version);
```

MemoryFree on StackVariable\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=17
Status	New

Calling free() (line 7) on a variable that was not dynamically allocated (line 7) in file flipperzero-firmware/mifare_desfire.c may result with a crash.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	25	25
Object	key_version	key_version

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method void mf_df_clear(MifareDesfireData* data) {

```
....
25.                     free(key_version);
```

MemoryFree on StackVariable\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=18
Status	New

Calling free() (line 7) on a variable that was not dynamically allocated (line 7) in file flipperzero-firmware/mifare_desfire.c may result with a crash.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	34	34
Object	file	file

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method void mf_df_clear(MifareDesfireData* data) {

```
....
34.                 free(file);
```

MemoryFree on StackVariable\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=19
Status	New

Calling free() (line 7) on a variable that was not dynamically allocated (line 7) in file flipperzero-firmware/mifare_desfire.c may result with a crash.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c

Line	37	37
Object	app	app

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method void mf_df_clear(MifareDesfireData* data) {

```
....
37.          free(app);
```

Memory Leak

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

Categories

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

[Description](#)

Memory Leak\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=41
Status	New

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	360	360
Object	app	app

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method bool mf_df_parse_get_application_ids_response(

```
....
360.          MifareDesfireApplication* app =
malloc(sizeof(MifareDesfireApplication));
```

Memory Leak\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=42
Status	New

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c

Line	395	395
Object	file	file

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method bool mf_df_parse_get_file_ids_response(uint8_t* buf, uint16_t len, MifareDesfireFile** file_head) {

```
....
395.            MifareDesfireFile* file =
malloc(sizeof(MifareDesfireFile));
```

Memory Leak\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=43
Status	New

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	491	491
Object	contents	contents

Code Snippet
File Name flipperzero-firmware/mifare_desfire.c
Method bool mf_df_parse_read_data_response(uint8_t* buf, uint16_t len, MifareDesfireFile* out) {

```
....
491.        out->contents = malloc(len);
```

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=1050072&projectid=50062&pathid=44
Status	New

The variable declared in pvReturn at flipperzero-firmware/memmgr_heap.c in line 339 is not initialized when it is used by pvReturn at flipperzero-firmware/memmgr_heap.c in line 339.

	Source	Destination
File	flipperzero-firmware/memmgr_heap.c	flipperzero-firmware/memmgr_heap.c
Line	341	482
Object	pvReturn	pvReturn

Code Snippet
File Name flipperzero-firmware/memmgr_heap.c
Method void* pvPortMalloc(size_t xWantedSize) {

```
....
341.        void* pvReturn = NULL;
....
482.        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=1050072&projectid=50062&pathid=45
Status	New

The variable declared in pxNextFreeBlock at flipperzero-firmware/memmgr_heap.c in line 339 is not initialized when it is used by print_heap_block at flipperzero-firmware/memmgr_heap.c in line 339.

	Source	Destination
File	flipperzero-firmware/memmgr_heap.c	flipperzero-firmware/memmgr_heap.c
Line	448	451
Object	pxNextFreeBlock	print_heap_block

Code Snippet
File Name flipperzero-firmware/memmgr_heap.c
Method void* pvPortMalloc(size_t xWantedSize) {

```
....
448.                    pxBlock->pxNextFreeBlock = NULL;
....
451.                    print_heap_block = pxBlock;
```

Use of Zero Initialized Pointer\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=46
Status	New

The variable declared in pxNextFreeBlock at flipperzero-firmware/memmgr_heap.c in line 568 is not initialized when it is used by pxNextFreeBlock at flipperzero-firmware/memmgr_heap.c in line 568.

	Source	Destination
File	flipperzero-firmware/memmgr_heap.c	flipperzero-firmware/memmgr_heap.c
Line	597	603
Object	pxNextFreeBlock	pxNextFreeBlock

Code Snippet
File Name flipperzero-firmware/memmgr_heap.c
Method static void prvHeapInit(void) {

```
....
597.       pxEnd->pxNextFreeBlock = NULL;
....
603.       pxFirstFreeBlock->pxNextFreeBlock = pxEnd;
```

Char Overflow

Query Path:
CPP\Cx\CPP Integer Overflow\Char Overflow Version:1

Categories

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

Description

Char Overflow\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=21
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 254 of flipperzero-firmware/memmgr_heap.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	flipperzero-firmware/memmgr_heap.c	flipperzero-firmware/memmgr_heap.c
Line	264	264
Object	AssignExpr	AssignExpr

Code Snippet
File Name flipperzero-firmware/memmgr_heap.c
Method char* ultoa(unsigned long num, char* str, int radix) {

```
....
264.             temp[temp_loc++] = digit + '0';
```

Char Overflow\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=22
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 254 of flipperzero-firmware/memmgr_heap.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	flipperzero-firmware/memmgr_heap.c	flipperzero-firmware/memmgr_heap.c
Line	266	266
Object	AssignExpr	AssignExpr

Code Snippet
File Name flipperzero-firmware/memmgr_heap.c
Method char* ultoa(unsigned long num, char* str, int radix) {

```
....
266.             temp[temp_loc++] = digit - 10 + 'A';
```

Unchecked Return Value

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

Categories

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

Description

Unchecked Return Value\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=1
Status	New

The mf_df_parse_read_data_response method calls the contents function, at line 485 of flipperzero-firmware/mifare_desfire.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	491	491
Object	contents	contents

Code Snippet

File Name	flipperzero-firmware/mifare_desfire.c
Method	bool mf_df_parse_read_data_response(uint8_t* buf, uint16_t len, MifareDesfireFile* out) {

```
....
491.        out->contents = malloc(len);
```

Unchecked Return Value\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=2
Status	New

The subghz_on_system_start method calls the arg function, at line 898 of flipperzero-firmware/subghz_cli.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	935	935
Object	arg	arg

Code Snippet

File Name	flipperzero-firmware/subghz_cli.c
Method	void subghz_on_system_start() {

```
....
935.            pb_region.bands.arg = malloc(sizeof(FuriHalRegion));
```

Unchecked Return Value\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=3
Status	New

The subghz_on_system_start_istream_decode_band method calls the region function, at line 864 of flipperzero-firmware/subghz_cli.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	878	878
Object	region	region

Code Snippet

File Name	flipperzero-firmware/subghz_cli.c
Method	static bool subghz_on_system_start_istream_decode_band(

```
....
878.        region = realloc( //-V701
```

Unchecked Return Value\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=4
Status	New

The mf_df_parse_get_application_ids_response method calls the app function, at line 347 of flipperzero-firmware/mifare_desfire.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	360	360
Object	app	app

Code Snippet

File Name	flipperzero-firmware/mifare_desfire.c
Method	bool mf_df_parse_get_application_ids_response(

```
....
360.            MifareDesfireApplication* app =
malloc(sizeof(MifareDesfireApplication));
```

Unchecked Return Value\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=5
Status	New

The mf_df_parse_get_file_ids_response method calls the file function, at line 388 of flipperzero-firmware/mifare_desfire.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	flipperzero-firmware/mifare_desfire.c	flipperzero-firmware/mifare_desfire.c
Line	395	395
Object	file	file

Code Snippet

File Name	flipperzero-firmware/mifare_desfire.c
Method	bool mf_df_parse_get_file_ids_response(uint8_t* buf, uint16_t len, MifareDesfireFile** file_head) {

```
....
395.            MifareDesfireFile* file =
malloc(sizeof(MifareDesfireFile));
```

Unchecked Return Value\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=6
Status	New

The protocol_keri_alloc method calls the protocol function, at line 37 of flipperzero-firmware/protocol_keri.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	flipperzero-firmware/protocol_keri.c	flipperzero-firmware/protocol_keri.c
Line	38	38
Object	protocol	protocol

Code Snippet

File Name	flipperzero-firmware/protocol_keri.c
Method	ProtocolKeri* protocol_keri_alloc(void) {

```
....
38.       ProtocolKeri* protocol = malloc(sizeof(ProtocolKeri));
```

Unchecked Return Value\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=7
Status	New

The protocol_nexwatch_alloc method calls the protocol function, at line 47 of flipperzero-firmware/protocol_nexwatch.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	flipperzero-firmware/protocol_nexwatch.c	flipperzero-firmware/protocol_nexwatch.c
Line	48	48
Object	protocol	protocol

Code Snippet
File Name flipperzero-firmware/protocol_nexwatch.c
Method ProtocolNexwatch* protocol_nexwatch_alloc(void) {

```
....
48.      ProtocolNexwatch* protocol = malloc(sizeof(ProtocolNexwatch));
```

Unchecked Return Value\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=8
Status	New

The subghz_cli_command_rx method calls the instance function, at line 233 of flipperzero-firmware/subghz_cli.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	253	253
Object	instance	instance

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method void subghz_cli_command_rx(Cli* cli, FuriString* args, void* context) {

```
....
253.      SubGhzCliCommandRx* instance =
malloc(sizeof(SubGhzCliCommandRx));
```

Unchecked Return Value\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=9
Status	New

The subghz_cli_command_rx_raw method calls the instance function, at line 317 of flipperzero-firmware/subghz_cli.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	337	337
Object	instance	instance

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method void subghz_cli_command_rx_raw(Cli* cli, FuriString* args, void* context) {

```
....
337.        SubGhzCliCommandRx* instance =
malloc(sizeof(SubGhzCliCommandRx));
```

Unchecked Return Value\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=10
Status	New

The subghz_cli_command_decode_raw method calls the instance function, at line 392 of flipperzero-firmware/subghz_cli.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	flipperzero-firmware/subghz_cli.c	flipperzero-firmware/subghz_cli.c
Line	442	442
Object	instance	instance

Code Snippet
File Name flipperzero-firmware/subghz_cli.c
Method void subghz_cli_command_decode_raw(Cli* cli, FuriString* args, void* context) {

```
....
442.            SubGhzCliCommandRx* instance =
malloc(sizeof(SubGhzCliCommandRx));
```

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=1050072&projectid=50062&pathid=20
Status	New

The variable declared in 0 at flipperzero-firmware/memmgr_heap.c in line 568 is not initialized when it is used by xStart at flipperzero-firmware/memmgr_heap.c in line 568.

	Source	Destination
File	flipperzero-firmware/memmgr_heap.c	flipperzero-firmware/memmgr_heap.c
Line	588	588
Object	0	xStart

Code Snippet
File Name flipperzero-firmware/memmgr_heap.c
Method static void prvHeapInit(void) {

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

Unchecked Array Index

Query Path:
CPP\Cx\CPP Low Visibility\Unchecked Array Index Version:1

Categories

NIST SP 800-53: SI-10 Information Input Validation (P1)

Description

Unchecked Array Index\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050072&projectid=50062&pathid=23
Status	New

	Source	Destination
File	flipperzero-firmware/memmgr_heap.c	flipperzero-firmware/memmgr_heap.c
Line	276	276
Object	str_loc	str_loc

Code Snippet
File Name flipperzero-firmware/memmgr_heap.c
Method char* ultoa(unsigned long num, char* str, int radix) {

```
....
276.      str[str_loc] = 0; // add null termination.
```

Buffer Overflow boundcpy WrongSizeParam

Risk

What might happen

Buffer overflow attacks, in their various forms, could allow an attacker to control certain areas of memory. Typically, this is used to overwrite data on the stack necessary for the program to function properly, such as code and memory addresses, though other forms of this attack exist. Exploiting this vulnerability can generally lead to system crashes, infinite loops, or even execution of arbitrary code.

Cause

How does it happen

Buffer Overflows can manifest in numerous different variations. In it's most basic form, the attack controls a buffer, which is then copied to a smaller buffer without size verification. Because the attacker's source buffer is larger than the program's target buffer, the attacker's data overwrites whatever is next on the stack, allowing the attacker to control program structures.
Alternatively, the vulnerability could be the result of improper bounds checking; exposing internal memory addresses outside of their valid scope; allowing the attacker to control the size of the target buffer; or various other forms.

General Recommendations

How to avoid it

- Always perform proper bounds checking before copying buffers or strings.
- Prefer to use safer functions and structures, e.g. safe string classes over `char*`, `strncpy` over `strcpy`, and so on.
- Consistently apply tests for the size of buffers.
- Do not return variable addresses outside the scope of their variables.

Source Code Examples

CPP

Overflowing Buffers

```
const int BUFFER_SIZE = 10;
char buffer[BUFFER_SIZE];

void copyStringToBuffer(char* inputString)
{
    strcpy(buffer, inputString);
}
```

Checked Buffers

```
const int BUFFER_SIZE = 10;
const int MAX_INPUT_SIZE = 256;
char buffer[BUFFER_SIZE];

void copyStringToBuffer(char* inputString)
{
    if (strnlen(inputString, MAX_INPUT_SIZE) < sizeof(buffer))
    {
        strncpy(buffer, inputString, sizeof(buffer));
```

```
    }
}
```

MemoryFree on StackVariable

Risk

What might happen

Undefined Behavior may result with a crash. Crashes may give an attacker valuable information about the system and the program internals. Furthermore, it may leave unprotected files (e.g memory) that may be exploited.

Cause

How does it happen

Calling free() on a variable that was not dynamically allocated (e.g. malloc) will result with an Undefined Behavior.

General Recommendations

How to avoid it

Use free() only on dynamically allocated variables in order to prevent unexpected behavior from the compiler.

Source Code Examples

CPP

Bad - Calling free() on a static variable

```
void clean_up(){
  char temp[256];
  do_something();
  free(tmp);
  return;
}
```

Good - Calling free() only on variables that were dynamically allocated

```
void clean_up(){
  char *buff;
  buff = (char*) malloc(1024);
  free(buff);
  return;
}
```

Char Overflow

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

Source Code Examples

CPP

Unsafe Downsize Casting

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

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

    return total;
}
```

Safer Use of Proper Data Types

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

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

    // check if total will overflow available integer size
    if (INT_MAX - abs(op2) > op1)
```

```
    {
        total = op1 + op2;
    }
    else
    {
        // instead of overflow, saturate (but this is not always a good thing)
        total = INT_MAX
    }

    return total;
}
```

Dangerous Functions

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

Source Code Examples

CPP

Buffer Overflow in gets()

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

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

Safe reading from user

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

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

Unsafe function for string copy

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

      return 0;
}
```

Safe string copy

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

      return 0;
}
```

Unsafe format string

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

Safe format string

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

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

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

Description

Description Summary

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

Extended Description

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

Terminology Notes

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

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

C

C++

Modes of Introduction

Memory leaks have two common and sometimes overlapping causes:

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

Common Consequences

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

Likelihood of Exploit

Medium

Demonstrative Examples

Example 1

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

(Bad Code)

Example Language: **C**

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

return NULL;
}
```

```
return buf;
}
```

Example 2

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

(Bad Code)

Example Language: **C**

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

free(foo);
}
int main() {

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

Observed Examples

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

Potential Mitigations

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

Phase: Architecture and Design

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

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

Relationships

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

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

Relationship Notes

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

Affected Resources

- Memory

Functional Areas

- Memory management

Taxonomy Mappings

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

White Box Definitions

A weakness where the code path has:

1. start statement that allocates dynamically allocated memory resource

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

Where "loses" is defined through the following scenarios:

1. identity of the dynamic allocated memory resource never obtained

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

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

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

References

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

Content History

Submissions

Submission Date	Submitter	Organization	Source
	PLOVER		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci	Cigital	External
	updated Time of Introduction		
2008-08-01		KDM Analytics	External
	added/updated white box definitions		
2008-08-15		Veracode	External
	Suggested OWASP Top Ten 2004 mapping		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Relationships, Other Notes, References, Relationship Notes, Taxonomy Mappings, Terminology Notes		
2008-10-14	CWE Content Team	MITRE	Internal
	updated Description		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Other Notes		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Name		
2009-07-17	KDM Analytics		External
	Improved the White Box Definition		

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

Previous Entry Names

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

BACK TO TOP

Use of Zero Initialized Pointer

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

Source Code Examples

CPP

Explicit NULL Dereference

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

Implicit NULL Dereference

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

Java

Explicit Null Dereference

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

Unchecked Return Value

Risk

What might happen

A program that does not check function return values could cause the application to enter an undefined state. This could lead to unexpected behavior and unintended consequences, including inconsistent data, system crashes or other error-based exploits.

Cause

How does it happen

The application calls a system function, but does not receive or check the result of this funciton. These functions often return error codes in the result, or share other status codes with it's caller. The application simply ignores this result value, losing this vital information.

General Recommendations

How to avoid it

- Always check the result of any called function that returns a value, and verify the result is an expected value.

- Ensure the calling function responds to all possible return values.

- Expect runtime errors and handle them gracefully. Explicitly define a mechanism for handling unexpected errors.

Source Code Examples

CPP

Unchecked Memory Allocation

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

Safer Memory Allocation

```
buff = (char*) malloc(size+1);
if (buff==NULL) exit(1);

strncpy(buff, source, size);
buff[size] = '\0';
```

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

Improper Validation of Array Index

Weakness ID: 129 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The product uses untrusted input when calculating or using an array index, but the product does not validate or incorrectly validates the index to ensure the index references a valid position within the array.

Alternate Terms

out-of-bounds array index

index-out-of-range

array index underflow

Time of Introduction

- Implementation

Applicable Platforms

Languages

C: *(Often)*

C++: *(Often)*

Language-independent

Common Consequences

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

Likelihood of Exploit

High

Detection Methods

Automated Static Analysis

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

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

Effectiveness: High

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

Automated Dynamic Analysis

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

Black Box

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

Demonstrative Examples

Example 1

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

(Bad Code)

Example Language: **C**

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

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

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

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

(Good Code)

Example Language: **C**

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

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

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

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

Example 2

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

(Bad Code)

Example Language: **Java**

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

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

Example 3

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

(Bad Code)

Example Language: **Java**

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

String productSummary = new String("");

try {
String productSummary = getProductSummary(index);

} catch (Exception ex) {...}

return productSummary;
}

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

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

(Good Code)

Example Language: **Java**

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

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

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

} catch (Exception ex) {...}

return productSummary;
}

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

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

return productSummary;
}
```

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

(Good Code)

Example Language: **Java**

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

Observed Examples

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

Potential Mitigations

Phase: Architecture and Design

Strategies: Input Validation; Libraries or Frameworks

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

Phase: Architecture and Design

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

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

Phase: Requirements

Strategy: Language Selection

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

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

Phase: Implementation

Strategy: Input Validation

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

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

Phase: Implementation

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

Weakness Ordinalities

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

Relationships

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

Theoretical Notes

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

Affected Resources

- Memory

f Causal Nature

Explicit

Taxonomy Mappings

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

Related Attack Patterns

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

References

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

Content History

Submissions

Submission Date	Submitter	Organization	Source
	CLASP		Externally Mined

Modifications

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

Previous Entry Names

Change Date	Previous Entry Name
2009-10-29	Unchecked Array Indexing

BACK TO TOP

Scanned Languages

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