thx-library-pio Scan Report

Project Name	thx-library-pio
Scan Start	Wednesday, October 04, 2017 11:26:26 AM
Preset	Checkmarx Default
Scan Time	00h:02m:01s
Lines Of Code Scanned	12229
Files Scanned	108
Report Creation Time	Wednesday, October 04, 2017 11:29:16 AM
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108
Team	CxServer
Checkmarx Version	8.4.1
Scan Type	Full
Source Origin	LocalPath
Density	1/100 (Vulnerabilities/LOC)
Visibility	Public

Filter Settings

Severity

Included: High, Medium, Low, Information

Excluded: None

Result State

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

Excluded: None

Assigned to

Included: All

Categories

Included:

Uncategorized	All
Custom	All
PCI DSS v3.1	All
OWASP Top 10 2013	All

Excluded:

Uncategorized	None
Custom	None
PCI DSS v3.1	None
OWASP Top 10 2013	None

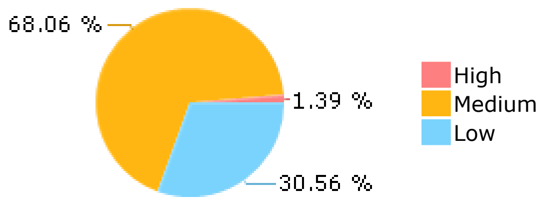
Results Limit

Results limit per query was set to 50

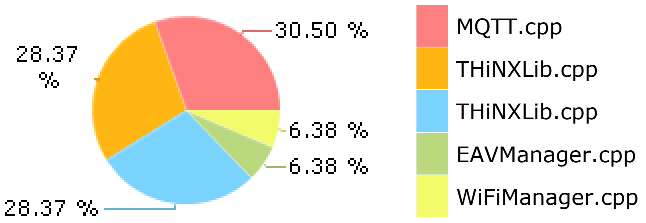
Selected Queries

Selected queries are listed in Result Summary

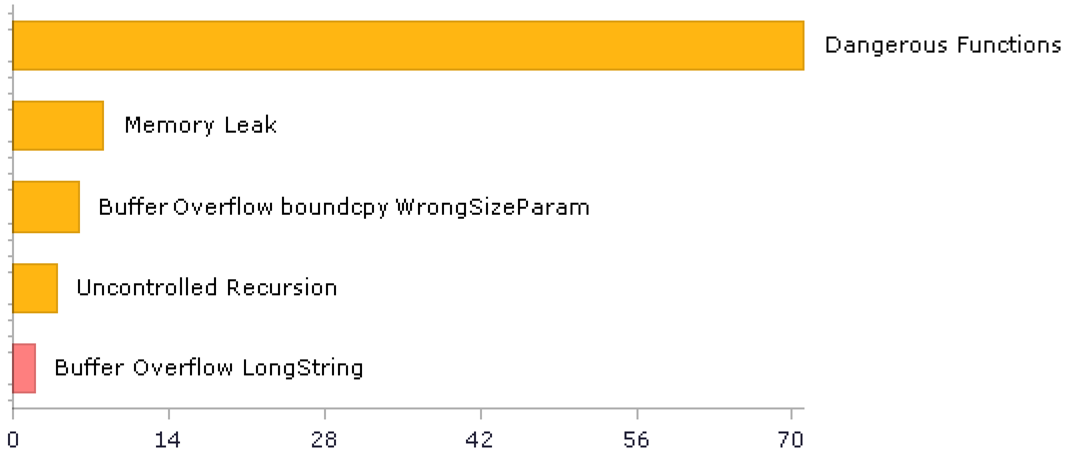
Result Summary



Most Vulnerable Files



Top 5 Vulnerabilities



Scan Summary - OWASP Top 10 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	2	2
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	71	71
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 v3.1

Further details and elaboration about vulnerabilities and risks can be found at: PCI DSS v3.1

Category	Issues Found	Best Fix Locations
PCI DSS (3.1) - 6.5.1 - Injection flaws - particularly SQL injection	0	0
PCI DSS (3.1) - 6.5.2 - Buffer overflows	8	8
PCI DSS (3.1) - 6.5.3 - Insecure cryptographic storage	0	0
PCI DSS (3.1) - 6.5.4 - Insecure communications	0	0
PCI DSS (3.1) - 6.5.5 - Improper error handling*	0	0
PCI DSS (3.1) - 6.5.7 - Cross-site scripting (XSS)	0	0
PCI DSS (3.1) - 6.5.8 - Improper access control	0	0
PCI DSS (3.1) - 6.5.9 - Cross-site request forgery	0	0
PCI DSS (3.1) - 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 - Custom

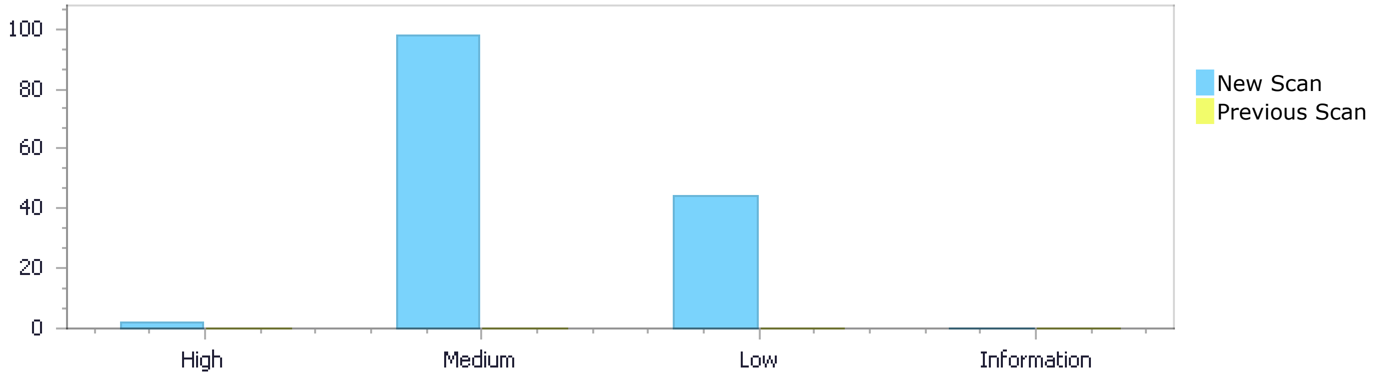
Category	Issues Found	Best Fix Locations
Must audit	0	0
Check	0	0
Optional	0	0

Results Distribution By Status

First scan of the project

	High	Medium	Low	Information	Total
New Issues	2	98	44	0	144
Recurrent Issues	0	0	0	0	0
Total	2	98	44	0	144

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	2	98	44	0	144
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	2	98	44	0	144

Result Summary

Vulnerability Type	Occurrences	Severity
Buffer Overflow LongString	2	High
Dangerous Functions	71	Medium
Memory Leak	8	Medium
Buffer Overflow boundcpy WrongSizeParam	6	Medium
Uncontrolled Recursion	4	Medium

Use of Uninitialized Variable	4	Medium
Heap Inspection	2	Medium
Stored Buffer Overflow boundcpy	2	Medium
Use After Free	1	Medium
Improper Resource Access Authorization	14	Low
Unchecked Array Index	11	Low
Unchecked Return Value	8	Low
Potential Precision Problem	4	Low
Information Exposure Through Comments	3	Low
Incorrect Permission Assignment For Critical Resources	2	Low
TOCTOU	2	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	33
/thinx-firmware-esp8266-pio/THiNXLib.cpp	33
/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	18
/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp	7
/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp	7
/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.h	1
/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/PubSubClient.cpp	1

Scan Results Details

Buffer Overflow LongString

Query Path:
CPP\Cx\CPP Buffer Overflow\Buffer Overflow LongString Version:1

Categories

PCI DSS v3.1: PCI DSS (3.1) - 6.5.2 - Buffer overflows

Description

Buffer Overflow LongString\Path 1:

Severity	High
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=1
Status	New

The size of the buffer used by EAVManager::handleWifi in parLength, at line 371 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that EAVManager::handleWifi passes to "%d", at line 371 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp, to overwrite the target buffer.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Line	468	469
Object	"%d"	parLength

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Method void EAVManager::handleWifi(boolean scan) {

```
....
468.        snprintf(parLength, 2, "%d", _params[i]->getValueLength());
469.        pitem.replace("{l}", parLength);
```

Buffer Overflow LongString\Path 2:

Severity	High
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=2
Status	New

The size of the buffer used by WiFiManager::handleWifi in parLength, at line 380 of /thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp, is not properly verified before writing data to the buffer.

This can enable a buffer overflow attack, using the source buffer that WiFiManager::handleWifi passes to "%d", at line 380 of /thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp, to overwrite the target buffer.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Line	477	478
Object	"%d"	parLength

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Method void WiFiManager::handleWifi(boolean scan) {

```
....
477.         snprintf(parLength, 2, "%d", _params[i]->getValueLength());
478.         pitem.replace("{l}", parLength);
```

Dangerous Functions

Query Path:
CPP\Cx\CPP Medium Threat\Dangerous Functions Version:0

Categories

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

Description

Dangerous Functions\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=13
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	298	298
Object	memcpy	memcpy

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method Connect& Connect::set_will(String willTopic, String willMessage, uint8_t willQos, bool willRetain) {

```
....
298.      memcpy(_will_message, willMessage.c_str(), _will_message_len);
```

Dangerous Functions\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=14
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	313	313
Object	memcpy	memcpy

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp

Method Connect& Connect::set_will(String willTopic, uint8_t *willMessage, uint16_t willMessageLength, uint8_t willQos, bool willRetain) {

```
....
313.      memcpy(_will_message, willMessage, _will_message_len);
```

Dangerous Functions\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=15
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	32	32
Object	memcpy	memcpy

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp

Method void write(uint8_t *buf, uint32_t& bufpos, uint8_t *data, uint16_t dlen) {

```
....
32.      memcpy(buf + bufpos, data, dlen);
```

Dangerous Functions\Path 4:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=16
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	50	50
Object	memcpy	memcpy

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp

Method void write_bare_payload(uint8_t *buf, uint32_t& bufpos, uint8_t *data, uint32_t dlen) {

```
....
50.     memcpy(buf + bufpos, data, dlen);
```

Dangerous Functions\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=17
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	405	405
Object	memcpy	memcpy

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp

Method Publish::Publish(String topic, String payload) :

```
....
405.        memcpy(_payload, payload.c_str(), payload.length());
```

Dangerous Functions\Path 6:

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

	18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=18
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	439	439
Object	memcpy	memcpy

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method Publish::Publish(uint8_t flags, uint8_t* data, uint32_t length) :

```
....
439.        memcpy(_payload, data + pos, _payload_len);
```

Dangerous Functions\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=19
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	654	654
Object	sprintf	sprintf

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::thinx_mqtt_channel() {

```
....
654.    sprintf(mqtt_device_channel, "/%s/%s", thinx_owner, thinx_udid);
```

Dangerous Functions\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=20
Status	New

	Source	Destination

File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	662	662
Object	sprintf	sprintf

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::thinx_mqtt_status_channel() {

```
....
662.     sprintf(mqtt_device_status_channel, "/%s/%s/status",
thinx_owner, thinx_udid);
```

Dangerous Functions\Path 9:

Severity Medium
Result State To Verify
Online Results http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=21
Status New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	672	672
Object	sprintf	sprintf

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method const char * THiNX::thinx_mac() {

```
....
672.   sprintf(mac_string, "5CCF7F%6X", ESP.getChipId()); // ESP8266
only!
```

Dangerous Functions\Path 10:

Severity Medium
Result State To Verify
Online Results http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=22
Status New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	646	646
Object	sprintf	sprintf

Code Snippet	
File Name	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Method	String THiNX::thinx_mqtt_channel() {

```
....
646.     sprintf(mqtt_device_channel, "/%s/%s", thinx_owner, thinx_udid);
```

Dangerous Functions\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=23
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	654	654
Object	sprintf	sprintf

Code Snippet	
File Name	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Method	String THiNX::thinx_mqtt_status_channel() {

```
....
654.     sprintf(mqtt_device_status_channel, "/%s/%s/status",
thinx_owner, thinx_udid);
```

Dangerous Functions\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=24
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	664	664
Object	sprintf	sprintf

Code Snippet	
File Name	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Method	const char * THiNX::thinx_mac() {

```
....
664.   sprintf(mac_string, "5CCF7F%6X", ESP.getChipId()); // ESP8266
only!
```

Dangerous Functions\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=25
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.h	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.h
Line	118	118
Object	strcpy	strcpy

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.h

Method inline void saveConfigCallback(void) {

```
....
118.        strcpy(thx_api_key, api_key_param->getValue());
```

Dangerous Functions\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=26
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	831	831
Object	strcpy	strcpy

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp

Method void THiNX::configCallback() {

```
....
831.    strcpy(thx_api_key, api_key_param->getValue());
```

Dangerous Functions\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=27
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	823	823
Object	strcpy	strcpy

Code Snippet

File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp

Method void THiNX::configCallback() {

```
....
823.    strcpy(thx_api_key, api_key_param->getValue());
```

Dangerous Functions\Path 16:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=28
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Line	83	83
Object	strlen	strlen

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp

Method void EAVManager::setupConfigPortal() {

```
....
83.      if (strlen(_apPassword) < 8 || strlen(_apPassword) > 63) {
```

Dangerous Functions\Path 17:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108

	&pathid=29
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Line	83	83
Object	strlen	strlen

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp

Method void EAVManager::setupConfigPortal() {

```
....
83.       if (strlen(_apPassword) < 8 || strlen(_apPassword) > 63) {
```

Dangerous Functions\Path 18:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=30
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	1293	1293
Object	strlen	strlen

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp

Method void THiNX::loop() {

```
....
1293.         if (strlen(thinx_api_key) > 4) {
```

Dangerous Functions\Path 19:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=31
Status	New

	Source	Destination

File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	125	125
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method THiNX::THiNX(const char * __apikey) {

```
....
125.     if (strlen(thinx_api_key) > 4) {
```

Dangerous Functions\Path 20:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=32
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	128	128
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method THiNX::THiNX(const char * __apikey) {

```
....
128.       if (strlen(__apikey) > 4) {
```

Dangerous Functions\Path 21:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=33
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	156	156
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method void THiNX::initWithAPIKey(const char * __apikey) {

```
....
156.     if (strlen(thinx_api_key) < 4) {
```

Dangerous Functions\Path 22:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=34
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	157	157
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method void THiNX::initWithAPIKey(const char * __apikey) {

```
....
157.       if (strlen(__apikey) > 1) {
```

Dangerous Functions\Path 23:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=35
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	300	300
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
300.      if (strlen(THINX_FIRMWARE_VERSION) > 1) {
```

Dangerous Functions\Path 24:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=36
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	304	304
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
304.      if (strlen(THINX_FIRMWARE_VERSION_SHORT) > 1) {
```

Dangerous Functions\Path 25:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=37
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	308	308
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
308.      if (strlen(THINX_COMMIT_ID) > 1) {
```

Dangerous Functions\Path 26:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=38

Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	312	312
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
312.      if (strlen(thinx_owner) > 1) {
```

Dangerous Functions\Path 27:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=39
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	316	316
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
316.      if (strlen(thinx_alias) > 1) {
```

Dangerous Functions\Path 28:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=40
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp

Line	320	320
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
320.      if (strlen(thinx_udid) > 4) {
```

Dangerous Functions\Path 29:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=41
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	484	484
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method void THiNX::parse(String payload) {

```
....
484.            if (strlen(available_update_url) > 5) {
```

Dangerous Functions\Path 30:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=42
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	557	557
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method void THiNX::parse(String payload) {

```
....
557.            if (strlen(available_update_url) > 4) {
```

Dangerous Functions\Path 31:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=43
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	569	569
Object	strlen	strlen

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method	void THiNX::parse(String payload) {

```
....
569.            if (strlen(available_update_url) > 4) {
```

Dangerous Functions\Path 32:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=44
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	687	687
Object	strlen	strlen

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method	void THiNX::publish() {

```
....
687.    if (strlen(thinx_udid) < 4) return;
```

Dangerous Functions\Path 33:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=45
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	730	730
Object	strlen	strlen

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp

Method bool THiNX::start_mqtt() {

```
....
730.    if (strlen(thinx_udid) < 4) {
```

Dangerous Functions\Path 34:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=46
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	745	745
Object	strlen	strlen

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp

Method bool THiNX::start_mqtt() {

```
....
745.    if (strlen(thinx_api_key) < 5) {
```

Dangerous Functions\Path 35:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=47
Status	New

	Source	Destination

File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	1017	1017
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method void THiNX::deviceInfo() {

```
....
1017.     if (strlen(thinx_owner) > 1) {
```

Dangerous Functions\Path 36:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=48
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	1023	1023
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method void THiNX::deviceInfo() {

```
....
1023.     if (strlen(thinx_api_key) > 1) {
```

Dangerous Functions\Path 37:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=49
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	1029	1029
Object	strlen	strlen

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method	void THiNX::deviceInfo() {

```
....
1029.     if (strlen(thinx_udid) > 1) {
```

Dangerous Functions\Path 38:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=50
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	1037	1037
Object	strlen	strlen

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method	void THiNX::deviceInfo() {

```
....
1037.     if (strlen(available_update_url) > 1) {
```

Dangerous Functions\Path 39:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=51
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	1129	1129
Object	strlen	strlen

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method	void THiNX::import_build_time_constants() {

```
....
1129.     if (strlen(thinx_api_key) < 4) {
```

Dangerous Functions\Path 40:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=52
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	1133	1133
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method void THiNX::import_build_time_constants() {

```
....
1133.     if (strlen(THINX_UDID) > 2) {
```

Dangerous Functions\Path 41:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=53
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	1183	1183
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method void THiNX::evt_save_api_key() {

```
....
1183.       if (strlen(thx_api_key) > 4) {
```

Dangerous Functions\Path 42:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=54

Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Line	81	81
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Method void WiFiManager::setupConfigPortal() {

```
....
81.      if (strlen(_apPassword) < 8 || strlen(_apPassword) > 63) {
```

Dangerous Functions\Path 43:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=55
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Line	81	81
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Method void WiFiManager::setupConfigPortal() {

```
....
81.      if (strlen(_apPassword) < 8 || strlen(_apPassword) > 63) {
```

Dangerous Functions\Path 44:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=56
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp

Line	1293	1293
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method void THiNX::loop() {

```
....
1293.         if (strlen(thinx_api_key) > 4) {
```

Dangerous Functions\Path 45:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=57
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	117	117
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method THiNX::THiNX(const char * __apikey) {

```
....
117.     if (strlen(thinx_api_key) > 4) {
```

Dangerous Functions\Path 46:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=58
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	120	120
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method THiNX::THiNX(const char * __apikey) {

```
....
120.        if (strlen(__apikey) > 4) {
```

Dangerous Functions\Path 47:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=59
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	148	148
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method void THiNX::initWithAPIKey(const char * __apikey) {

```
....
148.      if (strlen(thinx_api_key) < 4) {
```

Dangerous Functions\Path 48:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=60
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	149	149
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method void THiNX::initWithAPIKey(const char * __apikey) {

```
....
149.        if (strlen(__apikey) > 1) {
```

Dangerous Functions\Path 49:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=61
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	292	292
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
292.      if (strlen(THINX_FIRMWARE_VERSION) > 1) {
```

Dangerous Functions\Path 50:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=62
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	296	296
Object	strlen	strlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
296.      if (strlen(THINX_FIRMWARE_VERSION_SHORT) > 1) {
```

Memory Leak

Query Path:
CPP\Cx\CPP Medium Threat\Memory Leak Version:2

Description

Memory Leak\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108

	&pathid=86
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	301	301
Object	root	root

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
301.        root["firmware"] = strdup(THINX_FIRMWARE_VERSION);
```

Memory Leak\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=87
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	305	305
Object	root	root

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
305.        root["version"] = strdup(THINX_FIRMWARE_VERSION_SHORT);
```

Memory Leak\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=88
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp

Line	309	309
Object	root	root

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
309.          root["commit"] = strdup(THINX_COMMIT_ID);
```

Memory Leak\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=89
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	324	324
Object	root	root

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
324.      root["platform"] = strdup(THINX_PLATFORM);
```

Memory Leak\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=90
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	293	293
Object	root	root

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method String THiNX::checkin_body() {

```
....
293.        root["firmware"] = strdup(THINX_FIRMWARE_VERSION);
```

Memory Leak\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=91
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	297	297
Object	root	root

Code Snippet

File Name	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Method	String THiNX::checkin_body() {

```
....
297.        root["version"] = strdup(THINX_FIRMWARE_VERSION_SHORT);
```

Memory Leak\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=92
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	301	301
Object	root	root

Code Snippet

File Name	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Method	String THiNX::checkin_body() {

```
....
301.         root["commit"] = strdup(THINX_COMMIT_ID);
```

Memory Leak\Path 8:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=93
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	316	316
Object	root	root

Code Snippet

File Name	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Method	String THiNX::checkin_body() {

```
....
316.      root["platform"] = strdup(THINX_PLATFORM);
```

Buffer Overflow boundcpy WrongSizeParam

Query Path:
CPP\Cx\CPP Buffer Overflow\Buffer Overflow boundcpy WrongSizeParam Version:0

Categories

PCI DSS v3.1: PCI DSS (3.1) - 6.5.2 - Buffer overflows

Description

Buffer Overflow boundcpy WrongSizeParam\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=3
Status	New

The size of the buffer used by Connect::set_will in _will_message_len, at line 288 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that Connect::set_will passes to _will_message_len, at line 288 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, to overwrite the target buffer.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	298	298
Object	_will_message_len	_will_message_len

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp

Method	Connect& Connect::set_will(String willTopic, String willMessage, uint8_t willQos, bool willRetain) {

```
....
298.       memcpy(_will_message, willMessage.c_str(), _will_message_len);
```

Buffer Overflow boundcpy WrongSizeParam\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=4
Status	New

The size of the buffer used by Connect::set_will in _will_message_len, at line 303 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that Connect::set_will passes to _will_message_len, at line 303 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, to overwrite the target buffer.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	313	313
Object	_will_message_len	_will_message_len

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	Connect& Connect::set_will(String willTopic, uint8_t *willMessage, uint16_t willMessageLength, uint8_t willQos, bool willRetain) {

```
....
313.       memcpy(_will_message, willMessage, _will_message_len);
```

Buffer Overflow boundcpy WrongSizeParam\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=5
Status	New

The size of the buffer used by write in dlen, at line 30 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that write passes to dlen, at line 30 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, to overwrite the target buffer.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-

	arduino/src/PubSubClient/MQTT.cpp	arduino/src/PubSubClient/MQTT.cpp
Line	32	32
Object	dlen	dlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method void write(uint8_t *buf, uint32_t& bufpos, uint8_t *data, uint16_t dlen) {

```
....
32.       memcpy(buf + bufpos, data, dlen);
```

Buffer Overflow boundcpy WrongSizeParam\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=6
Status	New

The size of the buffer used by write_bare_payload in dlen, at line 49 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that write_bare_payload passes to dlen, at line 49 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, to overwrite the target buffer.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	50	50
Object	dlen	dlen

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method void write_bare_payload(uint8_t *buf, uint32_t& bufpos, uint8_t *data, uint32_t dlen) {

```
....
50.       memcpy(buf + bufpos, data, dlen);
```

Buffer Overflow boundcpy WrongSizeParam\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=7
Status	New

The size of the buffer used by EAVManagerParameter::init in length, at line 32 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that EAVManagerParameter::init passes to length, at line 32 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp, to overwrite the target buffer.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Line	41	41
Object	length	length

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Method	void EAVManagerParameter::init(const char *id, const char *placeholder, const char *defaultValue, int length, const char *custom) {

```
....
41.       strncpy(_value, defaultValue, length);
```

Buffer Overflow boundcpy WrongSizeParam\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=8
Status	New

The size of the buffer used by WiFiManagerParameter::init in length, at line 32 of /thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that WiFiManagerParameter::init passes to length, at line 32 of /thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp, to overwrite the target buffer.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Line	41	41
Object	length	length

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Method	void WiFiManagerParameter::init(const char *id, const char *placeholder, const char *defaultValue, int length, const char *custom) {

```
....
41.       strncpy(_value, defaultValue, length);
```

Use of Uninitialized Variable

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

Description

Use of Uninitialized Variable\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=95
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Line	258	270
Object	status	status

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Method	uint8_t EAVManager::waitForConnectResult() {

```
....
258.        uint8_t status;
....
270.        return status;
```

Use of Uninitialized Variable\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=96
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Line	267	279
Object	status	status

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Method	uint8_t WiFiManager::waitForConnectResult() {

```
....
267.        uint8_t status;
....
279.        return status;
```

Use of Uninitialized Variable\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=97
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Line	434	435
Object	rssiQ	rssiQ

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Method void EAVManager::handleWifi(boolean scan) {

```
....
434.                String rssiQ;
435.                rssiQ += quality;
```

Use of Uninitialized Variable\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=98
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Line	443	444
Object	rssiQ	rssiQ

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Method void WiFiManager::handleWifi(boolean scan) {

```
....
443.            String rssiQ;
444.            rssiQ += quality;
```

Uncontrolled Recursion

Query Path:
CPP\Cx\CPP Medium Threat\Uncontrolled Recursion Version:1

Description

Uncontrolled Recursion\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=115
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	72	72
Object	read	read

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	String read<String>(uint8_t *buf, uint32_t& pos) {

```
....
72.       uint16_t len = read<uint16_t>(buf, pos);
```

Uncontrolled Recursion\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=116
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	94	94
Object	read	read

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	uint16_t read<uint16_t>(Client& client) {

```
....
94.       val |= read<uint8_t>(client);
```

Uncontrolled Recursion\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=117
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	93	93
Object	read	read

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	uint16_t read<uint16_t>(Client& client) {

```
....
93.       uint16_t val = read<uint8_t>(client) << 8;
```

Uncontrolled Recursion\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=118
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	100	100
Object	read	read

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	String read<String>(Client& client) {

```
....
100.      uint16_t len = read<uint16_t>(client);
```

Heap Inspection

Query Path:
CPP\Cx\CPP Medium Threat\Heap Inspection Version:1

Categories

OWASP Top 10 2013: A6-Sensitive Data Exposure

Description

Heap Inspection\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=84
Status	New

Method THiNX::start_mqtt at line 720 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp defines pass, which is designated to contain user passwords. However, while plaintext passwords are later assigned to pass, this variable is never cleared from memory.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	757	757
Object	pass	pass

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method bool THiNX::start_mqtt() {

```
....
757.    const char* pass = thinx_api_key;
```

Heap Inspection\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=85
Status	New

Method THiNX::start_mqtt at line 712 of /thinx-firmware-esp8266-pio/THiNXLib.cpp defines pass, which is designated to contain user passwords. However, while plaintext passwords are later assigned to pass, this variable is never cleared from memory.

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp

Line	749	749
Object	pass	pass

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method bool THiNX::start_mqtt() {

```
....
749.     const char* pass = thinx_api_key;
```

Stored Buffer Overflow boundcpy

Query Path:
CPP\Cx\CPP Stored Vulnerabilities\Stored Buffer Overflow boundcpy Version:1

Description

Stored Buffer Overflow boundcpy\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=99
Status	New

The size of the buffer used by Publish::Publish in _payload_len, at line 426 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that Publish::Publish passes to pos, at line 426 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, to overwrite the target buffer.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	432	439
Object	pos	_payload_len

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method Publish::Publish(uint8_t flags, uint8_t* data, uint32_t length) :

```
....
432.       _topic = read<String>(data, pos);
....
439.         memcpy(_payload, data + pos, _payload_len);
```

Stored Buffer Overflow boundcpy\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=100
Status	New

The size of the buffer used by Publish::Publish in _payload_len, at line 426 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that Publish::Publish passes to pos, at line 426 of /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp, to overwrite the target buffer.

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	434	439
Object	pos	_payload_len

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	Publish::Publish(uint8_t flags, uint8_t* data, uint32_t length) :

```
....
434.         _packet_id = read<uint16_t>(data, pos);
....
439.         memcpy(_payload, data + pos, _payload_len);
```

Use After Free

Query Path:

CPP\Cx\CPP Medium Threat\Use After Free Version:0

Description

Use After Free\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=94
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/PubSubClient.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/PubSubClient.cpp
Line	137	136
Object	msg	msg

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/PubSubClient.cpp
Method	bool PubSubClient::_wait_for(MQTT::message_type match_type, uint16_t match_pid) {

```
....
137.                delete msg;
....
136.                uint16_t pid = msg->packet_id();
```

Improper Resource Access Authorization

Query Path:
CPP\Cx\CPP Low Visibility\Improper Resource Access Authorization Version:1

Description

Improper Resource Access Authorization\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=101
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	390	390
Object	pos	pos

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method ConnectAck::ConnectAck(uint8_t* data, uint32_t length) :

```
....
390.       uint8_t reserved = read<uint8_t>(data, pos);
```

Improper Resource Access Authorization\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=102
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	392	392
Object	pos	pos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	ConnectAck::ConnectAck(uint8_t* data, uint32_t length) :

```
....
392.        _rc = read<uint8_t>(data, pos);
```

Improper Resource Access Authorization\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=103
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	72	72
Object	pos	pos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	String read<String>(uint8_t *buf, uint32_t& pos) {

```
....
72.       uint16_t len = read<uint16_t>(buf, pos);
```

Improper Resource Access Authorization\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=104
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	76	76
Object	pos	pos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	String read<String>(uint8_t *buf, uint32_t& pos) {

```
....
76.         val += (char)read<uint8_t>(buf, pos);
```

Improper Resource Access Authorization\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=105
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	219	219
Object	rem	rem

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	Message* readPacket(Client& client) {

```
....
219.          int read_size = client.read(read_point, rem);
```

Improper Resource Access Authorization\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=106
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	432	432
Object	pos	pos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	Publish::Publish(uint8_t flags, uint8_t* data, uint32_t length) :

```
....
432.        _topic = read<String>(data, pos);
```

Improper Resource Access Authorization\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=107
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	434	434
Object	pos	pos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	Publish::Publish(uint8_t flags, uint8_t* data, uint32_t length) :

```
....
434.          _packet_id = read<uint16_t>(data, pos);
```

Improper Resource Access Authorization\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=108
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	541	541
Object	pos	pos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	PublishAck::PublishAck(uint8_t* data, uint32_t length) :

```
....
541.        _packet_id = read<uint16_t>(data, pos);
```

Improper Resource Access Authorization\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=109
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	556	556
Object	pos	pos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	PublishRec::PublishRec(uint8_t* data, uint32_t length) :

```
....
556.        _packet_id = read<uint16_t>(data, pos);
```

Improper Resource Access Authorization\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=110
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	579	579
Object	pos	pos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	PublishRel::PublishRel(uint8_t* data, uint32_t length) :

```
....
579.        _packet_id = read<uint16_t>(data, pos);
```

Improper Resource Access Authorization\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=111
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	602	602
Object	pos	pos

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method PublishComp::PublishComp(uint8_t* data, uint32_t length) :

```
....
602.        _packet_id = read<uint16_t>(data, pos);
```

Improper Resource Access Authorization\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=112
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	667	667
Object	pos	pos

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method SubscribeAck::SubscribeAck(uint8_t* data, uint32_t length) :

```
....
667.        _packet_id = read<uint16_t>(data, pos);
```

Improper Resource Access Authorization\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=113
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	673	673
Object	pos	pos

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method SubscribeAck::SubscribeAck(uint8_t* data, uint32_t length) :

```
....
673.          _rcs[i] = read<uint8_t>(data, pos);
```

Improper Resource Access Authorization\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=114
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	751	751
Object	pos	pos

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method UnsubscribeAck::UnsubscribeAck(uint8_t* data, uint32_t length) :

```
....
751.        _packet_id = read<uint16_t>(data, pos);
```

Unchecked Array Index

Query Path:
CPP\Cx\CPP Low Visibility\Unchecked Array Index Version:0

Description

Unchecked Array Index\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=127
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	123	123
Object	bufpos	bufpos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	void Message::write_fixed_header(uint8_t *buf, uint32_t& bufpos, uint32_t rlength) const {

```
....
123.        buf[bufpos] = _type << 4;
```

Unchecked Array Index\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=128
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	127	127
Object	bufpos	bufpos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	void Message::write_fixed_header(uint8_t *buf, uint32_t& bufpos, uint32_t rlength) const {

```
....
127.        buf[bufpos] |= _flags & 0x0f;
```

Unchecked Array Index\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=129
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	132	132
Object	bufpos	bufpos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	void Message::write_fixed_header(uint8_t *buf, uint32_t& bufpos, uint32_t rlength) const {

```
....
132.        buf[bufpos] |= 0x02;
```

Unchecked Array Index\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=130
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	331	331
Object	bufpos	bufpos

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp

Method void Connect::write_variable_header(uint8_t *buf, uint32_t& bufpos) const {

```
....
331.       buf[bufpos] = 0;          // Connect flags
```

Unchecked Array Index\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=131
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	333	333
Object	bufpos	bufpos

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp

Method void Connect::write_variable_header(uint8_t *buf, uint32_t& bufpos) const {

```
....
333.         buf[bufpos] |= 0x02;
```

Unchecked Array Index\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=132
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	336	336
Object	bufpos	bufpos

Code Snippet

File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp

Method void Connect::write_variable_header(uint8_t *buf, uint32_t& bufpos) const {

```
....
336.        buf[bufpos] |= 0x04;
```

Unchecked Array Index\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=133
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	339	339
Object	bufpos	bufpos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	void Connect::write_variable_header(uint8_t *buf, uint32_t& bufpos) const {

```
....
339.        buf[bufpos] |= 2 << 3;
```

Unchecked Array Index\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=134
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	341	341
Object	bufpos	bufpos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	void Connect::write_variable_header(uint8_t *buf, uint32_t& bufpos) const {

```
....
341.        buf[bufpos] |= _will_qos << 3;
```

Unchecked Array Index\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=135
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	342	342
Object	bufpos	bufpos

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method void Connect::write_variable_header(uint8_t *buf, uint32_t& bufpos) const {

```
....
342.        buf[bufpos] |= _will_retain << 5;
```

Unchecked Array Index\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=136
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	346	346
Object	bufpos	bufpos

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method void Connect::write_variable_header(uint8_t *buf, uint32_t& bufpos) const {

```
....
346.        buf[bufpos] |= 0x80;
```

Unchecked Array Index\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=137
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Line	348	348
Object	bufpos	bufpos

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/PubSubClient/MQTT.cpp
Method	void Connect::write_variable_header(uint8_t *buf, uint32_t& bufpos) const {

```
....
348.        buf[bufpos] |= 0x40;
```

Unchecked Return Value

Query Path:

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

Description

Unchecked Return Value\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=119
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Line	468	468
Object	snprintf	snprintf

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Method	void EAVManager::handleWifi(boolean scan) {

```
....
468.        snprintf(parLength, 2, "%d", _params[i]->getValueLength());
```

Unchecked Return Value\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=120
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	654	654
Object	sprintf	sprintf

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method	String THiNX::thinx_mqtt_channel() {

```
....
654.    sprintf(mqtt_device_channel, "/%s/%s", thinx_owner, thinx_udid);
```

Unchecked Return Value\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=121
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	662	662
Object	sprintf	sprintf

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method	String THiNX::thinx_mqtt_status_channel() {

```
....
662.    sprintf(mqtt_device_status_channel, "/%s/%s/status",
thinx_owner, thinx_udid);
```

Unchecked Return Value\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=122
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	672	672
Object	sprintf	sprintf

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method	const char * THiNX::thinx_mac() {

```
....
672.   sprintf(mac_string, "5CCF7F%6X", ESP.getChipId()); // ESP8266
only!
```

Unchecked Return Value\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=123
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Line	477	477
Object	snprintf	snprintf

Code Snippet

File Name	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Method	void WiFiManager::handleWifi(boolean scan) {

```
....
477.        snprintf(parLength, 2, "%d", _params[i]->getValueLength());
```

Unchecked Return Value\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=124
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	646	646
Object	sprintf	sprintf

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method String THiNX::thinx_mqtt_channel() {

```
....
646.    sprintf(mqtt_device_channel, "/%s/%s", thinx_owner, thinx_udid);
```

Unchecked Return Value\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=125
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	654	654
Object	sprintf	sprintf

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method String THiNX::thinx_mqtt_status_channel() {

```
....
654.    sprintf(mqtt_device_status_channel, "/%s/%s/status",
thinx_owner, thinx_udid);
```

Unchecked Return Value\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=126

Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	664	664
Object	sprintf	sprintf

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method const char * THiNX::thinx_mac() {

```
....
664.    sprintf(mac_string, "5CCF7F%6X", ESP.getChipId()); // ESP8266
only!
```

Potential Precision Problem

Query Path:
CPP\Cx\CPP Buffer Overflow\Potential Precision Problem Version:0

Description

Potential Precision Problem\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=9
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	654	654
Object	"/%s/%s"	"/%s/%s"

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::thinx_mqtt_channel() {

```
....
654.     sprintf(mqtt_device_channel, "/%s/%s", thinx_owner, thinx_udid);
```

Potential Precision Problem\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=10
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	662	662
Object	"/%s/%s/status"	"/%s/%s/status"

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method String THiNX::thinx_mqtt_status_channel() {

```
....
662.     sprintf(mqtt_device_status_channel, "/%s/%s/status",
thinx_owner, thinx_udid);
```

Potential Precision Problem\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=11
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	646	646
Object	"/%s/%s"	"/%s/%s"

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method String THiNX::thinx_mqtt_channel() {

```
....
646.     sprintf(mqtt_device_channel, "/%s/%s", thinx_owner, thinx_udid);
```

Potential Precision Problem\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=12
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	654	654

Object	"/%s/%s/status"	"/%s/%s/status"

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method String THiNX::thinx_mqtt_status_channel() {

```
....
654.     sprintf(mqtt_device_status_channel, "/%s/%s/status",
thinx_owner, thinx_udid);
```

Information Exposure Through Comments

Query Path:
CPP\Cx\CPP Low Visibility\Information Exposure Through Comments Version:1

Description

Information Exposure Through Comments\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=142
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Line	98	98
Object	_apPassword	_apPassword

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/EAVManager/EAVManager.cpp
Method void EAVManager::setupConfigPortal() {

```
....
98.      WiFi.softAP(_apName, _apPassword);//password option
```

Information Exposure Through Comments\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=143
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/thinx.h	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/thinx.h

Line	24	24
Object	"1234567890"	"1234567890"

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/thinx.h
Method

```
....
24.
```

Information Exposure Through Comments\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=144
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp	/thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Line	96	96
Object	_apPassword	_apPassword

Code Snippet
File Name /thinx-firmware-esp8266-pio/lib/WiFiManager/WiFiManager.cpp
Method void WiFiManager::setupConfigPortal() {

```
....
96.       WiFi.softAP(_apName, _apPassword);//password option
```

Incorrect Permission Assignment For Critical Resources

Query Path:
CPP\Cx\CPP Low Visibility\Incorrect Permission Assignment For Critical Resources Version:1

Description

Incorrect Permission Assignment For Critical Resources\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=138
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	988	988
Object	f	f

Code Snippet	
File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Method	void THiNX::save_device_info()

```
....
988.      File f = SPIFFS.open("/thx.cfg", "w");
```

Incorrect Permission Assignment For Critical Resources\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=139
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	988	988
Object	f	f

Code Snippet	
File Name	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Method	void THiNX::save_device_info()

```
....
988.      File f = SPIFFS.open("/thx.cfg", "w");
```

TOCTOU

Query Path:
CPP\Cx\CPP Low Visibility\TOCTOU Version:1

Description

TOCTOU\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=140
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp
Line	988	988
Object	open	open

Code Snippet	
File Name	/thinx-firmware-esp8266-pio/lib/thinx-lib-esp8266-arduino/src/THiNXLib.cpp

Method void THiNX::save_device_info()

```
....
988.      File f = SPIFFS.open("/thx.cfg", "w");
```

TOCTOU\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-18IMTI68O0K/CxWebClient/ViewerMain.aspx?scanid=1000113&projectid=108&pathid=141
Status	New

	Source	Destination
File	/thinx-firmware-esp8266-pio/THiNXLib.cpp	/thinx-firmware-esp8266-pio/THiNXLib.cpp
Line	988	988
Object	open	open

Code Snippet
File Name /thinx-firmware-esp8266-pio/THiNXLib.cpp
Method void THiNX::save_device_info()

```
....
988.      File f = SPIFFS.open("/thx.cfg", "w");
```

Buffer Overflow LongString

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

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

Use of Inherently Dangerous Function

Weakness ID: 242 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The program calls a function that can never be guaranteed to work safely.

Extended Description

Certain functions behave in dangerous ways regardless of how they are used. Functions in this category were often implemented without taking security concerns into account. The gets() function is unsafe because it does not perform bounds checking on the size of its input. An attacker can easily send arbitrarily-sized input to gets() and overflow the destination buffer. Similarly, the >> operator is unsafe to use when reading into a statically-allocated character array because it does not perform bounds checking on the size of its input. An attacker can easily send arbitrarily-sized input to the >> operator and overflow the destination buffer.

Time of Introduction

- Implementation

Applicable Platforms

Languages

C

C++

Likelihood of Exploit

High

Demonstrative Examples

Example 1

The excerpt below calls the gets() function in C, which is inherently unsafe.

(Bad Code)

Example Language: **C**

```
char buf[BUFSIZE];
gets(buf);
```

Example 2

The excerpt below calls the gets() function in C, which is inherently unsafe.

(Bad Code)

Example Language: **C**

```
char buf[24];
printf("Please enter your name and press <Enter>\n");
gets(buf);
...
}
```

However, the programmer uses the function gets() which is inherently unsafe because it blindly copies all input from STDIN to the buffer without checking size. This allows the user to provide a string that is larger than the buffer size, resulting in an overflow condition.

Potential Mitigations

Ban the use of dangerous function. Use their safe equivalent.

Use grep or static analysis tools to spot usage of dangerous functions.

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	Weakness Class	227	Failure to Fulfill API Contract ('API Abuse')	**Development Concepts (primary)699** **Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Class	710	Coding Standards Violation	**Research Concepts (primary)1000**
ChildOf	Category	748	CERT C Secure Coding Section 50 - POSIX (POS)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
CanPrecede	Weakness Base	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	Research Concepts1000

f Causal Nature

Explicit

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Dangerous Functions
CERT C Secure Coding	POS33-C		Do not use vfork()

References

Herbert Schildt. "Herb Schildt's C++ Programming Cookbook". Chapter 5. Working with I/O. McGraw-Hill Osborne Media. 2008-04-28.

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 5, "gets and fgets" Page 163. 2nd Edition. Microsoft. 2002.

Content History

Submissions

Submission Date	Submitter	Organization	Source
	7 Pernicious Kingdoms		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Sean Eidemiller	Cigital	External
	added/updated demonstrative examples		
2008-07-01	Eric Dalci	Cigital	External
	updated Potential Mitigations		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Relationships, Other Notes, Taxonomy Mappings, Type, Weakness Ordinalities		
2008-11-24	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Description, Other Notes, References		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples, References, Relationships		
2010-04-05	CWE Content Team	MITRE	Internal
	updated Relationships		

Previous Entry Names

Change Date	Previous Entry Name
2008-01-30	Dangerous Functions
2008-04-11	Use of Inherently Dangerous Functions

BACK TO TOP

Heap Inspection

Risk

What might happen

All variables stored by the application in unencrypted memory can potentially be retrieved by an unauthorized user, with privlieged access to the machine. For example, a privileged attacker could attach a debugger to the running process, or retrieve the process's memory from the swapfile or crash dump file.
Once the attacker finds the user passwords in memory, these can be reused to easily impersonate the user to the system.

Cause

How does it happen

String variables are immutable - in other words, once a string variable is assigned, its value cannot be changed or removed. Thus, these strings may remain around in memory, possibly in multiple locations, for an indefinite period of time until the garbage collector happens to remove it. Sensitive data, such as passwords, will remain exposed in memory as plaintext with no control over their lifetime.

General Recommendations

How to avoid it

Generic Guidance:

- Do not store senstiive data, such as passwords or encryption keys, in memory in plaintext, even for a short period of time.
- Prefer to use specialized classes that store encrypted memory.
- Alternatively, store secrets temporarily in mutable data types, such as byte arrays, and then promptly zeroize the memory locations.

Specific Recommendations - Java:

- Instead of storing passwords in immutable strings, prefer to use an encrypted memory object, such as SealedObject.

Specific Recommendations - .NET:

- Instead of storing passwords in immutable strings, prefer to use an encrypted memory object, such as SecureString or ProtectedData.

Source Code Examples

Java

Plaintext Password in Immutable String

```
class Heap_Inspection
{
  private string password;
  void setPassword()
  {
      password = System.console().readLine("Enter your password: ");
   }
}
```

Password Protected in Memory

```
class Heap_Inspection_Fixed
{
   private SealedObject password;
   void setPassword()
  {
      byte[] sKey = getKeyFromConfig();
      Cipher c = Cipher.getInstance("AES");
      c.init(Cipher.ENCRYPT_MODE, sKey);

      char[] input = System.console().readPassword("Enter your password: ");
      password = new SealedObject(Arrays.asList(input), c);
   }
}
```

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 Lifetime	**Research Concepts (primary)1000**

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

Relationship Notes

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

Affected Resources

- Memory

Functional Areas

- Memory management

Taxonomy Mappings

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

White Box Definitions

A weakness where the code path has:

1. start statement that allocates dynamically allocated memory resource

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

Where "loses" is defined through the following scenarios:

1. identity of the dynamic allocated memory resource never obtained

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

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

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

References

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

Content History

Submissions

Submission Date	Submitter	Organization	Source
	PLOVER		Externally Mined

Modifications

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

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

Previous Entry Names

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

BACK TO TOP

Use After Free

Weakness ID: 416 *(Weakness Base)* **Status:** Draft

Description

Description Summary

Referencing memory after it has been freed can cause a program to crash, use unexpected values, or execute code.

Alternate Terms

Use-After-Free

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

C

C++

Common Consequences

Scope	Effect
Integrity	The use of previously freed memory may corrupt valid data, if the memory area in question has been allocated and used properly elsewhere.
Availability	If chunk consolidation occur after the use of previously freed data, the process may crash when invalid data is used as chunk information.
Integrity	If malicious data is entered before chunk consolidation can take place, it may be possible to take advantage of a write-what-where primitive to execute arbitrary code.

Likelihood of Exploit

High

Demonstrative Examples

Example 1

(Bad Code)

Example Language: ***C***

```
#include <stdio.h>
#include <unistd.h>
#define BUFSIZER1 512
#define BUFSIZER2 ((BUFSIZER1/2) - 8)
int main(int argc, char **argv) {
char *buf1R1;
char *buf2R1;
char *buf2R2;
char *buf3R2;
buf1R1 = (char *) malloc(BUFSIZER1);
buf2R1 = (char *) malloc(BUFSIZER1);
free(buf2R1);
buf2R2 = (char *) malloc(BUFSIZER2);
buf3R2 = (char *) malloc(BUFSIZER2);
strncpy(buf2R1, argv[1], BUFSIZER1-1);
free(buf1R1);
free(buf2R2);
free(buf3R2);
}
```

Example 2

The following code illustrates a use after free error:

(Bad Code)

Example Language: **C**

```
char* ptr = (char*)malloc (SIZE); ...
if (err) {

abrt = 1;
free(ptr);
}
...
if (abrt) {
logError("operation aborted before commit", ptr);
}
```

Observed Examples

Reference	Description
CVE-2006-4997	freed pointer dereference

Potential Mitigations

Phase: Architecture and Design

Choose a language that provides automatic memory management.

Phase: Implementation

Ensuring that all pointers are set to NULL once they memory they point to has been freed can be an effective strategy. The utilization of multiple or complex data structures may lower the usefulness of this strategy.

Phase: Implementation

Use a static analysis tool to find instances of use after free.

Other Notes

The use of previously freed memory can have any number of adverse consequences -- ranging from the corruption of valid data to the execution of arbitrary code, depending on the instantiation and timing of the flaw. The simplest way data corruption may occur involves the system's reuse of the freed memory. Like double free errors and memory leaks, use after free errors 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. In this scenario, the memory in question is allocated to another pointer validly at some point after it has been freed. The original pointer to the freed memory is used again and points to somewhere within the new allocation. As the data is changed, it corrupts the validly used memory; this induces undefined behavior in the process. If the newly allocated data chances to hold a class, in C++ for example, various function pointers may be scattered within the heap data. If one of these function pointers is overwritten with an address to valid shellcode, execution of arbitrary code can be achieved.

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	Weakness Base	672	Operation on a Resource after Expiration or Release	**Research Concepts (primary)1000**
ChildOf	Category	742	CERT C Secure Coding Section 08 - Memory Management (MEM)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
ChildOf	Category	808	2010 Top 25 - Weaknesses On the Cusp	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**
CanPrecede	Weakness Base	120	Buffer Copy without	Research Concepts1000

			Checking Size of Input ('Classic Buffer Overflow')	
CanPrecede	Weakness Base	123	Write-what-where Condition	Research Concepts1000
MemberOf	View	630	Weaknesses Examined by SAMATE	**Weaknesses Examined by SAMATE (primary)630**
PeerOf	Weakness Base	364	Signal Handler Race Condition	Research Concepts1000
PeerOf	Weakness Variant	415	Double Free	Development Concepts699 Research Concepts1000

Affected Resources

- Memory

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Use After Free
CLASP			Using freed memory
CERT C Secure Coding	MEM00-C		Allocate and free memory in the same module, at the same level of abstraction
CERT C Secure Coding	MEM01-C		Store a new value in pointers immediately after free()
CERT C Secure Coding	MEM30-C		Do not access freed memory

White Box Definitions

A weakness where code path has:

1. start statement that relinquishes a dynamically allocated memory resource
2. end statement that accesses the dynamically allocated memory resource

Content History

Submissions

Submission Date	Submitter	Organization	Source
	7 Pernicious Kingdoms		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci	Cigital	External
	updated Potential Mitigations, 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, Observed Example, Other Notes, Taxonomy Mappings		
2008-11-24	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Common Consequences		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Relationships		

BACK TO TOP

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 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

Stored Buffer Overflow boundcpy

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));
    }
}
```

Uncontrolled Recursion

Weakness ID: 674 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The product does not properly control the amount of recursion that takes place, which consumes excessive resources, such as allocated memory or the program stack.

Alternate Terms

Stack Exhaustion

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

All

Common Consequences

Scope	Effect
Availability	Resources including CPU, memory, and stack memory could be rapidly consumed or exhausted, eventually leading to an exit or crash.
Confidentiality	In some cases, an application's interpreter might kill a process or thread that appears to be consuming too much resources, such as with PHP's memory_limit setting. When the interpreter kills the process/thread, it might report an error containing detailed information such as the application's installation path.

Observed Examples

Reference	Description
CVE-2007-1285	Deeply nested arrays trigger stack exhaustion.
CVE-2007-3409	Self-referencing pointers create infinite loop and resultant stack exhaustion.

Potential Mitigations

Limit the number of recursive calls to a reasonable number.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	361	Time and State	**Development Concepts (primary)699**
ChildOf	Weakness Class	691	Insufficient Control Flow Management	**Research Concepts (primary)1000**
ChildOf	Category	730	OWASP Top Ten 2004 Category A9 - Denial of Service	**Weaknesses in OWASP Top Ten (2004) (primary)711**

Affected Resources

- CPU

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
OWASP Top Ten 2004	A9	CWE More Specific	Denial of Service

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
82	Violating Implicit Assumptions Regarding XML Content (aka XML Denial of Service (XDoS))	
99	XML Parser Attack	

Content History

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci	Cigital	External
	updated Potential Mitigations, Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Common Consequences, Relationships, Taxonomy Mappings		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Related Attack Patterns		

BACK TO TOP

Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')

Weakness ID: 120 *(Weakness Base)* **Status:** Incomplete

Description

Description Summary

The program copies an input buffer to an output buffer without verifying that the size of the input buffer is less than the size of the output buffer, leading to a buffer overflow.

Extended Description

A buffer overflow condition exists when a program attempts to put more data in a buffer than it can hold, or when a program attempts to put data in a memory area outside of the boundaries of a buffer. The simplest type of error, and the most common cause of buffer overflows, is the "classic" case in which the program copies the buffer without checking its length at all. Other variants exist, but the existence of a classic overflow strongly suggests that the programmer is not considering even the most basic of security protections.

Alternate Terms

buffer overrun:	Some prominent vendors and researchers use the term "buffer overrun," but most people use "buffer overflow."
Unbounded Transfer	

Terminology Notes

Many issues that are now called "buffer overflows" are substantively different than the "classic" overflow, including entirely different bug types that rely on overflow exploit techniques, such as integer signedness errors, integer overflows, and format string bugs. This imprecise terminology can make it difficult to determine which variant is being reported.

Time of Introduction

- Implementation

Applicable Platforms

Languages

C

C++

Assembly

Common Consequences

Scope	Effect
Integrity	**Technical Impact:** *Execute unauthorized code or commands* Buffer overflows often can be used to execute arbitrary code, which is usually outside the scope of a program's implicit security policy. This can often be used to subvert any other security service.
Availability	Buffer overflows generally lead to crashes. Other attacks leading to lack of availability are possible, including putting the program into an infinite loop.

Likelihood of Exploit

High to Very 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 buffer overflows that originate from command line arguments in a program that is not expected to run with setuid or other special privileges.

Effectiveness: High

Detection techniques for buffer-related errors are more mature than for most other weakness types.

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.

Manual Analysis

Manual analysis can be useful for finding this weakness, but it might not achieve desired code coverage within limited time constraints. This becomes difficult for weaknesses that must be considered for all inputs, since the attack surface can be too large.

Demonstrative Examples

Example 1

The following code asks the user to enter their last name and then attempts to store the value entered in the last_name array.

(Bad Code)

Example Language: **C**

```
char last_name[20];
printf ("Enter your last name: ");
scanf ("%s", last_name);
```

The problem with the code above is that it does not check the size of the name entered by the user. If the user enters "Very_very_long_last_name" which is 24 characters long, then a buffer overflow will occur since the array can only hold 20 characters total.

Example 2

The following code attempts to create a local copy of a buffer to perform some manipulations to the data.

(Bad Code)

Example Language: **C**

```
void manipulate_string(char* string){
char buf[24];
strcpy(buf, string);
...
}
```

However, the programmer does not ensure that the size of the data pointed to by string will fit in the local buffer and blindly copies the data with the potentially dangerous strcpy() function. This may result in a buffer overflow condition if an attacker can influence the contents of the string parameter.

Example 3

The excerpt below calls the gets() function in C, which is inherently unsafe.

(Bad Code)

Example Language: **C**

```
char buf[24];
printf("Please enter your name and press <Enter>\n");
gets(buf);
...
}
```

However, the programmer uses the function gets() which is inherently unsafe because it blindly copies all input from STDIN to the buffer without checking size. This allows the user to provide a string that is larger than the buffer size, resulting in an overflow condition.

Example 4

In the following example, a server accepts connections from a client and processes the client request. After accepting a client connection, the program will obtain client information using the gethostbyaddr method, copy the hostname of the client that connected to a local variable and output the hostname of the client to a log file.

(Bad Code)

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

```
...
struct hostent *clienthp;
char hostname[MAX_LEN];

// create server socket, bind to server address and listen on socket
...

// accept client connections and process requests
int count = 0;
for (count = 0; count < MAX_CONNECTIONS; count++) {

int clientlen = sizeof(struct sockaddr_in);
int clientsocket = accept(serversocket, (struct sockaddr *)&clientaddr, &clientlen);

if (clientsocket >= 0) {
clienthp = gethostbyaddr((char *)&clientaddr.sin_addr.s_addr,
sizeof(clientaddr.sin_addr.s_addr), AF_INET);
strcpy(hostname, clienthp->h_name);
logOutput("Accepted client connection from host ", hostname);

// process client request
...
close(clientsocket);
}
}
close(serversocket);
...
```

However, the hostname of the client that connected may be longer than the allocated size for the local hostname variable. This will result in a buffer overflow when copying the client hostname to the local variable using the strcpy method.

Observed Examples

Reference	Description
CVE-2000-1094	buffer overflow using command with long argument
CVE-1999-0046	buffer overflow in local program using long environment variable
CVE-2002-1337	buffer overflow in comment characters, when product increments a counter for a ">" but does not decrement for "<"
CVE-2003-0595	By replacing a valid cookie value with an extremely long string of characters, an attacker may overflow the application's buffers.
CVE-2001-0191	By replacing a valid cookie value with an extremely long string of characters, an attacker may overflow the application's buffers.

Potential Mitigations

Phase: Requirements

Strategy: Language Selection

Use a language with features that can automatically mitigate or eliminate buffer overflows.

For example, many languages that perform their own memory management, such as Java and Perl, are not subject to buffer overflows. Other languages, such as Ada and C#, typically provide overflow protection, but the protection can be disabled by the programmer.

Be wary that a language's interface to native code may still be subject to overflows, even if the language itself is theoretically safe.

Phase: Architecture and Design

Strategy: Libraries or Frameworks

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

Examples include the Safe C String Library (SafeStr) by Messier and Viega, and the Strsafe.h library from Microsoft. These libraries provide safer versions of overflow-prone string-handling functions. This is not a complete solution, since many buffer overflows are not related to strings.

Phase: Build and Compilation

Run or compile your software using features or extensions that automatically provide a protection mechanism that mitigates or eliminates buffer overflows.

For example, certain compilers and extensions provide automatic buffer overflow detection mechanisms that are built into the compiled code. Examples include the Microsoft Visual Studio /GS flag, Fedora/Red Hat FORTIFY_SOURCE GCC flag, StackGuard, and ProPolice.

This is not necessarily a complete solution, since these mechanisms can only detect certain types of overflows. In addition, a buffer overflow attack can still cause a denial of service, since the typical response is to exit the application.

Phase: Implementation

Programmers should adhere to the following rules when allocating and managing their applications memory:

- Double check that your buffer is as large as you specify.
- When using functions that accept a number of bytes to copy, such as strncpy(), be aware that if the destination buffer size is equal to the source buffer size, it may not NULL-terminate the string.
- Check buffer boundaries if calling this function in a loop and make sure you are not in danger of writing past the allocated space.
- If necessary, truncate all input strings to a reasonable length before passing them to the copy and concatenation functions.

Phase: Operation

Use a feature like Address Space Layout Randomization (ASLR). This is not a complete solution. However, it forces the attacker to guess an unknown value that changes every program execution.

Phase: Operation

Use a CPU and operating system that offers Data Execution Protection (NX) or its equivalent. This is not a complete solution, since buffer overflows could be used to overwrite nearby variables to modify the software's state in dangerous ways. In addition, it cannot be used in cases in which self-modifying code is required.

Phases: Build and Compilation; Operation

Most mitigating technologies at the compiler or OS level to date address only a subset of buffer overflow problems and rarely provide complete protection against even that subset. It is good practice to implement strategies to increase the workload of an attacker, such as leaving the attacker to guess an unknown value that changes every program execution.

Phase: Implementation

Replace unbounded copy functions with analogous functions that support length arguments, such as strcpy with strncpy. Create these if they are not available.

Effectiveness: Moderate

This approach is still susceptible to calculation errors, including issues such as off-by-one errors (CWE-193) and incorrectly calculating buffer lengths (CWE-131).

Weakness Ordinalities

Ordinality	Description
Resultant	*(where the weakness is typically related to the presence of some other weaknesses)*
Primary	*(where the weakness exists independent of other weaknesses)*

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	20	Improper Input Validation	**Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Class	119	Failure to Constrain Operations within the Bounds of a Memory Buffer	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ChildOf	Category	633	Weaknesses that Affect Memory	**Resource-specific Weaknesses**

				(primary)631
ChildOf	Category	722	OWASP Top Ten 2004 Category A1 - Unvalidated Input	Weaknesses in OWASP Top Ten (2004)711
ChildOf	Category	726	OWASP Top Ten 2004 Category A5 - Buffer Overflows	**Weaknesses in OWASP Top Ten (2004) (primary)711**
ChildOf	Category	741	CERT C Secure Coding Section 07 - Characters and Strings (STR)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
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 Base	123	Write-what-where Condition	Research Concepts1000
ParentOf	Weakness Variant	785	Use of Path Manipulation Function without Maximum-sized Buffer	**Development Concepts (primary)699** Research Concepts1000
CanFollow	Weakness Base	170	Improper Null Termination	Research Concepts1000
CanFollow	Weakness Base	231	Improper Handling of Extra Values	Research Concepts1000
CanFollow	Weakness Base	242	Use of Inherently Dangerous Function	Research Concepts1000
CanFollow	Weakness Base	416	Use After Free	Research Concepts1000
CanFollow	Weakness Base	456	Missing Initialization	Research Concepts1000
PeerOf	Weakness Base	124	Buffer Underwrite ('Buffer Underflow')	Research Concepts1000
CanAlsoBe	Weakness Variant	196	Unsigned to Signed Conversion Error	Research Concepts1000

Relationship Notes

At the code level, stack-based and heap-based overflows do not differ significantly, so there usually is not a need to distinguish them. From the attacker perspective, they can be quite different, since different techniques are required to exploit them.

Affected Resources

- Memory

Functional Areas

- Memory Management

f Causal Nature

Explicit

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Unbounded Transfer ('classic overflow')
7 Pernicious Kingdoms			Buffer Overflow
CLASP			Buffer overflow
OWASP Top Ten 2004	A1	CWE More Specific	Unvalidated Input
OWASP Top Ten 2004	A5	CWE More Specific	Buffer Overflows
CERT C Secure Coding	STR35-C		Do not copy data from an unbounded source to a fixed-length array
WASC	7		Buffer Overflow

Related Attack Patterns

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

8	Buffer Overflow in an API Call
9	Buffer Overflow in Local Command-Line Utilities
10	Buffer Overflow via Environment Variables
14	Client-side Injection-induced Buffer Overflow
24	Filter Failure through Buffer Overflow
92	Forced Integer Overflow
42	MIME Conversion
44	Overflow Binary Resource File
45	Buffer Overflow via Symbolic Links
100	Overflow Buffers
46	Overflow Variables and Tags
47	Buffer Overflow via Parameter Expansion
67	String Format Overflow in syslog()

White Box Definitions

A weakness where the code path includes a Buffer Write Operation such that:

1. the expected size of the buffer is greater than the actual size of the buffer where expected size is equal to the sum of the size of the data item and the position in the buffer

Where Buffer Write Operation is a statement that writes a data item of a certain size into a buffer at a certain position and at a certain index

References

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 5, "Public Enemy #1: The Buffer Overrun" Page 127. 2nd Edition. Microsoft. 2002.

[REF-17] Michael Howard, David LeBlanc and John Viega. "24 Deadly Sins of Software Security". "Sin 5: Buffer Overruns." Page 89. McGraw-Hill. 2010.

Microsoft. "Using the Strsafe.h Functions". <http://msdn.microsoft.com/en-us/library/ms647466.aspx>.

Matt Messier and John Viega. "Safe C String Library v1.0.3". <http://www.zork.org/safestr/>.

Michael Howard. "Address Space Layout Randomization in Windows Vista". <http://blogs.msdn.com/michael_howard/archive/2006/05/26/address-space-layout-randomization-in-windows-vista.aspx>.

Arjan van de Ven. "Limiting buffer overflows with ExecShield". <http://www.redhat.com/magazine/009jul05/features/execshield/>.

"PaX". <http://en.wikipedia.org/wiki/PaX>.

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 Alternate Terms, Applicable Platforms, Common Consequences, Relationships, Observed Example, Other Notes, Taxonomy Mappings, Weakness Ordinalities		
2008-10-10	CWE Content Team	MITRE	Internal
	Changed name and description to more clearly emphasize the "classic" nature of the overflow.		
2008-10-14	CWE Content Team	MITRE	Internal
	updated Alternate Terms, Description, Name, Other Notes, Terminology Notes		

2008-11-24	CWE Content Team	MITRE	Internal
	updated Other Notes, Relationships, Taxonomy Mappings		
2009-01-12	CWE Content Team	MITRE	Internal
	updated Common Consequences, Other Notes, Potential Mitigations, References, Relationship Notes, Relationships		
2009-07-27	CWE Content Team	MITRE	Internal
	updated Other Notes, Potential Mitigations, Relationships		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Common Consequences, Relationships		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Demonstrative Examples, Detection Factors, Potential Mitigations, References, Related Attack Patterns, Relationships, Taxonomy Mappings, Time of Introduction, Type		
2010-04-05	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples, Related Attack Patterns		

Previous Entry Names

Change Date	Previous Entry Name
2008-10-14	Unbounded Transfer ('Classic Buffer Overflow')

BACK TO TOP

Improper Access Control (Authorization)

Weakness ID: 285 *(Weakness Class)* **Status:** Draft

Description

Description Summary

The software does not perform or incorrectly performs access control checks across all potential execution paths.

Extended Description

When access control checks are not applied consistently - or not at all - users are able to access data or perform actions that they should not be allowed to perform. This can lead to a wide range of problems, including information leaks, denial of service, and arbitrary code execution.

Alternate Terms

AuthZ:	"AuthZ" is typically used as an abbreviation of "authorization" within the web application security community. It is also distinct from "AuthC," which is an abbreviation of "authentication." The use of "Auth" as an abbreviation is discouraged, since it could be used for either authentication or authorization.

Time of Introduction

- Architecture and Design
- Implementation
- Operation

Applicable Platforms

Languages

Language-independent

Technology Classes

Web-Server: *(Often)*

Database-Server: *(Often)*

Modes of Introduction

A developer may introduce authorization weaknesses because of a lack of understanding about the underlying technologies. For example, a developer may assume that attackers cannot modify certain inputs such as headers or cookies.

Authorization weaknesses may arise when a single-user application is ported to a multi-user environment.

Common Consequences

Scope	Effect
Confidentiality	An attacker could read sensitive data, either by reading the data directly from a data store that is not properly restricted, or by accessing insufficiently-protected, privileged functionality to read the data.
Integrity	An attacker could modify sensitive data, either by writing the data directly to a data store that is not properly restricted, or by accessing insufficiently-protected, privileged functionality to write the data.
Integrity	An attacker could gain privileges by modifying or reading critical data directly, or by accessing insufficiently-protected, privileged functionality.

Likelihood of Exploit

High

Detection Methods

Automated Static Analysis

Automated static analysis is useful for detecting commonly-used idioms for authorization. A tool may be able to analyze related configuration files, such as .htaccess in Apache web servers, or detect the usage of commonly-used authorization libraries.

Generally, automated static analysis tools have difficulty detecting custom authorization schemes. In addition, the software's design may include some functionality that is accessible to any user and does not require an authorization check; an automated technique that detects the absence of authorization may report false positives.

Effectiveness: Limited

Automated Dynamic Analysis

Automated dynamic analysis may find many or all possible interfaces that do not require authorization, but manual analysis is required to determine if the lack of authorization violates business logic

Manual Analysis

This weakness can be detected using tools and techniques that require manual (human) analysis, such as penetration testing, threat modeling, and interactive tools that allow the tester to record and modify an active session.

Specifically, manual static analysis is useful for evaluating the correctness of custom authorization mechanisms.

Effectiveness: Moderate

These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules. However, manual efforts might not achieve desired code coverage within limited time constraints.

Demonstrative Examples

Example 1

The following program could be part of a bulletin board system that allows users to send private messages to each other. This program intends to authenticate the user before deciding whether a private message should be displayed. Assume that LookupMessageObject() ensures that the $id argument is numeric, constructs a filename based on that id, and reads the message details from that file. Also assume that the program stores all private messages for all users in the same directory.

(Bad Code)

Example Language: **Perl**

```
sub DisplayPrivateMessage {
my($id) = @_;
my $Message = LookupMessageObject($id);
print "From: " . encodeHTML($Message->{from}) . "<br>\n";
print "Subject: " . encodeHTML($Message->{subject}) . "\n";
print "<hr>\n";
print "Body: " . encodeHTML($Message->{body}) . "\n";
}

my $q = new CGI;
# For purposes of this example, assume that CWE-309 and
# CWE-523 do not apply.
if (! AuthenticateUser($q->param('username'), $q->param('password'))) {
ExitError("invalid username or password");
}

my $id = $q->param('id');
DisplayPrivateMessage($id);
```

While the program properly exits if authentication fails, it does not ensure that the message is addressed to the user. As a result, an authenticated attacker could provide any arbitrary identifier and read private messages that were intended for other users.

One way to avoid this problem would be to ensure that the "to" field in the message object matches the username of the authenticated user.

Observed Examples

Reference	Description
CVE-2009-3168	Web application does not restrict access to admin scripts, allowing authenticated users to reset administrative passwords.
CVE-2009-2960	Web application does not restrict access to admin scripts,

	allowing authenticated users to modify passwords of other users.
CVE-2009-3597	Web application stores database file under the web root with insufficient access control (CWE-219), allowing direct request.
CVE-2009-2282	Terminal server does not check authorization for guest access.
CVE-2009-3230	Database server does not use appropriate privileges for certain sensitive operations.
CVE-2009-2213	Gateway uses default "Allow" configuration for its authorization settings.
CVE-2009-0034	Chain: product does not properly interpret a configuration option for a system group, allowing users to gain privileges.
CVE-2008-6123	Chain: SNMP product does not properly parse a configuration option for which hosts are allowed to connect, allowing unauthorized IP addresses to connect.
CVE-2008-5027	System monitoring software allows users to bypass authorization by creating custom forms.
CVE-2008-7109	Chain: reliance on client-side security (CWE-602) allows attackers to bypass authorization using a custom client.
CVE-2008-3424	Chain: product does not properly handle wildcards in an authorization policy list, allowing unintended access.
CVE-2009-3781	Content management system does not check access permissions for private files, allowing others to view those files.
CVE-2008-4577	ACL-based protection mechanism treats negative access rights as if they are positive, allowing bypass of intended restrictions.
CVE-2008-6548	Product does not check the ACL of a page accessed using an "include" directive, allowing attackers to read unauthorized files.
CVE-2007-2925	Default ACL list for a DNS server does not set certain ACLs, allowing unauthorized DNS queries.
CVE-2006-6679	Product relies on the X-Forwarded-For HTTP header for authorization, allowing unintended access by spoofing the header.
CVE-2005-3623	OS kernel does not check for a certain privilege before setting ACLs for files.
CVE-2005-2801	Chain: file-system code performs an incorrect comparison (CWE-697), preventing defauls ACLs from being properly applied.
CVE-2001-1155	Chain: product does not properly check the result of a reverse DNS lookup because of operator precedence (CWE-783), allowing bypass of DNS-based access restrictions.

Potential Mitigations

Phase: Architecture and Design

Divide your application into anonymous, normal, privileged, and administrative areas. Reduce the attack surface by carefully mapping roles with data and functionality. Use role-based access control (RBAC) to enforce the roles at the appropriate boundaries.

Note that this approach may not protect against horizontal authorization, i.e., it will not protect a user from attacking others with the same role.

Phase: Architecture and Design

Ensure that you perform access control checks related to your business logic. These checks may be different than the access control checks that you apply to more generic resources such as files, connections, processes, memory, and database records. For example, a database may restrict access for medical records to a specific database user, but each record might only be intended to be accessible to the patient and the patient's doctor.

Phase: Architecture and Design

Strategy: Libraries or Frameworks

Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

For example, consider using authorization frameworks such as the JAAS Authorization Framework and the OWASP ESAPI Access

Control feature.

Phase: Architecture and Design

For web applications, make sure that the access control mechanism is enforced correctly at the server side on every page. Users should not be able to access any unauthorized functionality or information by simply requesting direct access to that page.

One way to do this is to ensure that all pages containing sensitive information are not cached, and that all such pages restrict access to requests that are accompanied by an active and authenticated session token associated with a user who has the required permissions to access that page.

Phases: System Configuration; Installation

Use the access control capabilities of your operating system and server environment and define your access control lists accordingly. Use a "default deny" policy when defining these ACLs.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	254	Security Features	**Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Class	284	Access Control (Authorization) Issues	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ChildOf	Category	721	OWASP Top Ten 2007 Category A10 - Failure to Restrict URL Access	**Weaknesses in OWASP Top Ten (2007) (primary)629**
ChildOf	Category	723	OWASP Top Ten 2004 Category A2 - Broken Access Control	**Weaknesses in OWASP Top Ten (2004) (primary)711**
ChildOf	Category	753	2009 Top 25 - Porous Defenses	**Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)750**
ChildOf	Category	803	2010 Top 25 - Porous Defenses	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**
ParentOf	Weakness Variant	219	Sensitive Data Under Web Root	**Research Concepts (primary)1000**
ParentOf	Weakness Base	551	Incorrect Behavior Order: Authorization Before Parsing and Canonicalization	**Development Concepts (primary)699** Research Concepts1000
ParentOf	Weakness Class	638	Failure to Use Complete Mediation	Research Concepts1000
ParentOf	Weakness Base	804	Guessable CAPTCHA	**Development Concepts (primary)699** **Research Concepts (primary)1000**

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Missing Access Control
OWASP Top Ten 2007	A10	CWE More Specific	Failure to Restrict URL Access
OWASP Top Ten 2004	A2	CWE More Specific	Broken Access Control

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
1	Accessing Functionality Not Properly Constrained by ACLs	
13	Subverting Environment Variable Values	
17	Accessing, Modifying or Executing Executable Files	
87	Forceful Browsing	

39	Manipulating Opaque Client-based Data Tokens
45	Buffer Overflow via Symbolic Links
51	Poison Web Service Registry
59	Session Credential Falsification through Prediction
60	Reusing Session IDs (aka Session Replay)
77	Manipulating User-Controlled Variables
76	Manipulating Input to File System Calls
104	Cross Zone Scripting

References

NIST. "Role Based Access Control and Role Based Security". <http://csrc.nist.gov/groups/SNS/rbac/>.

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 4, "Authorization" Page 114; Chapter 6, "Determining Appropriate Access Control" Page 171. 2nd Edition. Microsoft. 2002.

Content History

Submissions

Submission Date	Submitter	Organization	Source
	7 Pernicious Kingdoms		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci	Cigital	External
	updated Time of Introduction		
2008-08-15		Veracode	External
	Suggested OWASP Top Ten 2004 mapping		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Relationships, Other Notes, Taxonomy Mappings		
2009-01-12	CWE Content Team	MITRE	Internal
	updated Common Consequences, Description, Likelihood of Exploit, Name, Other Notes, Potential Mitigations, References, Relationships		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Potential Mitigations		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Description, Related Attack Patterns		
2009-07-27	CWE Content Team	MITRE	Internal
	updated Relationships		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Type		
2009-12-28	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Demonstrative Examples, Detection Factors, Modes of Introduction, Observed Examples, Relationships		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Alternate Terms, Detection Factors, Potential Mitigations, References, Relationships		
2010-04-05	CWE Content Team	MITRE	Internal
	updated Potential Mitigations		

Previous Entry Names

Change Date	Previous Entry Name
2009-01-12	Missing or Inconsistent Access Control

BACK TO TOP

Unchecked Return Value

Weakness ID: 252 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The software does not check the return value from a method or function, which can prevent it from detecting unexpected states and conditions.

Extended Description

Two common programmer assumptions are "this function call can never fail" and "it doesn't matter if this function call fails". If an attacker can force the function to fail or otherwise return a value that is not expected, then the subsequent program logic could lead to a vulnerability, because the software is not in a state that the programmer assumes. For example, if the program calls a function to drop privileges but does not check the return code to ensure that privileges were successfully dropped, then the program will continue to operate with the higher privileges.

Time of Introduction

- Implementation

Applicable Platforms

Languages

All

Common Consequences

Scope	Effect
Integrity	The data which were produced as a result of a function call could be in a bad state upon return. If the return value is not checked, then this bad data may be used in operations and lead to a crash or other unintended behaviors.

Likelihood of Exploit

Low

Demonstrative Examples

Example 1

Consider the following code segment:

(Bad Code)

Example Language: **C**

```
char buf[10], cp_buf[10];
fgets(buf, 10, stdin);
strcpy(cp_buf, buf);
```

The programmer expects that when fgets() returns, buf will contain a null-terminated string of length 9 or less. But if an I/O error occurs, fgets() will not null-terminate buf. Furthermore, if the end of the file is reached before any characters are read, fgets() returns without writing anything to buf. In both of these situations, fgets() signals that something unusual has happened by returning NULL, but in this code, the warning will not be noticed. The lack of a null terminator in buf can result in a buffer overflow in the subsequent call to strcpy().

Example 2

The following code does not check to see if memory allocation succeeded before attempting to use the pointer returned by malloc().

(Bad Code)

Example Language: **C**

```
buf = (char*) malloc(req_size);
```

```
strncpy(buf, xfer, req_size);
```

The traditional defense of this coding error is: "If my program runs out of memory, it will fail. It doesn't matter whether I handle the error or simply allow the program to die with a segmentation fault when it tries to dereference the null pointer." This argument ignores three important considerations:

- Depending upon the type and size of the application, it may be possible to free memory that is being used elsewhere so that execution can continue.
- It is impossible for the program to perform a graceful exit if required. If the program is performing an atomic operation, it can leave the system in an inconsistent state.
- The programmer has lost the opportunity to record diagnostic information. Did the call to malloc() fail because req_size was too large or because there were too many requests being handled at the same time? Or was it caused by a memory leak that has built up over time? Without handling the error, there is no way to know.

Example 3

The following code loops through a set of users, reading a private data file for each user. The programmer assumes that the files are always 1 kilobyte in size and therefore ignores the return value from Read(). If an attacker can create a smaller file, the program will recycle the remainder of the data from the previous user and handle it as though it belongs to the attacker.

(Bad Code)

Example Language: **Java**

```
char[] byteArray = new char[1024];
for (IEnumerator i=users.GetEnumerator(); i.MoveNext() ;i.Current()) {
String userName = (String) i.Current();
String pFileName = PFILE_ROOT + "/" + userName;
StreamReader sr = new StreamReader(pFileName);
sr.Read(byteArray,0,1024);//the file is always 1k bytes
sr.Close();
processPFile(userName, byteArray);
}
```

(Bad Code)

Example Language: **Java**

```
FileInputStream fis;
byte[] byteArray = new byte[1024];
for (Iterator i=users.iterator(); i.hasNext();) {
String userName = (String) i.next();
String pFileName = PFILE_ROOT + "/" + userName;
FileInputStream fis = new FileInputStream(pFileName);
fis.read(byteArray); // the file is always 1k bytes
fis.close();
processPFile(userName, byteArray);
```

Example 4

The following code does not check to see if the string returned by getParameter() is null before calling the member function compareTo(), potentially causing a NULL dereference.

(Bad Code)

Example Language: **Java**

```
String itemName = request.getParameter(ITEM_NAME);
if (itemName.compareTo(IMPORTANT_ITEM)) {
...
}
...
```

The following code does not check to see if the string returned by theItem property is null before calling the member function Equals(), potentially causing a NULL dereference. string itemName = request.Item(ITEM_NAME);

(Bad Code)

```
if (itemName.Equals(IMPORTANT_ITEM)) {
...
}
...
```

The traditional defense of this coding error is: "I know the requested value will always exist because.... If it does not exist, the program cannot perform the desired behavior so it doesn't matter whether I handle the error or simply allow the program to die dereferencing a null value." But attackers are skilled at finding unexpected paths through programs, particularly when exceptions are involved.

Example 5

The following code shows a system property that is set to null and later dereferenced by a programmer who mistakenly assumes it will always be defined.

(Bad Code)

```
System.clearProperty("os.name");
...
String os = System.getProperty("os.name");
if (os.equalsIgnoreCase("Windows 95")) System.out.println("Not supported");
```

The traditional defense of this coding error is: "I know the requested value will always exist because.... If it does not exist, the program cannot perform the desired behavior so it doesn't matter whether I handle the error or simply allow the program to die dereferencing a null value." But attackers are skilled at finding unexpected paths through programs, particularly when exceptions are involved.

Example 6

The following VB.NET code does not check to make sure that it has read 50 bytes from myfile.txt. This can cause DoDangerousOperation() to operate on an unexpected value.

(Bad Code)

```
Dim MyFile As New FileStream("myfile.txt", FileMode.Open, FileAccess.Read, FileShare.Read)
Dim MyArray(50) As Byte
MyFile.Read(MyArray, 0, 50)
DoDangerousOperation(MyArray(20))
```

In .NET, it is not uncommon for programmers to misunderstand Read() and related methods that are part of many System.IO classes. The stream and reader classes do not consider it to be unusual or exceptional if only a small amount of data becomes available. These classes simply add the small amount of data to the return buffer, and set the return value to the number of bytes or characters read. There is no guarantee that the amount of data returned is equal to the amount of data requested.

Example 7

It is not uncommon for Java programmers to misunderstand read() and related methods that are part of many java.io classes. Most errors and unusual events in Java result in an exception being thrown. But the stream and reader classes do not consider it unusual or exceptional if only a small amount of data becomes available. These classes simply add the small amount of data to the return buffer, and set the return value to the number of bytes or characters read. There is no guarantee that the amount of data returned is equal to the amount of data requested. This behavior makes it important for programmers to examine the return value from read() and other IO methods to ensure that they receive the amount of data they expect.

Example 8

This example takes an IP address from a user, verifies that it is well formed and then looks up the hostname and copies it into a buffer.

(Bad Code)

Example Language: **C**

```
void host_lookup(char *user_supplied_addr){
struct hostent *hp;
in_addr_t *addr;
char hostname[64];
in_addr_t inet_addr(const char *cp);

/*routine that ensures user_supplied_addr is in the right format for conversion */
validate_addr_form(user_supplied_addr);
addr = inet_addr(user_supplied_addr);
hp = gethostbyaddr( addr, sizeof(struct in_addr), AF_INET);
strcpy(hostname, hp->h_name);
}
```

If an attacker provides an address that appears to be well-formed, but the address does not resolve to a hostname, then the call to gethostbyaddr() will return NULL. When this occurs, a NULL pointer dereference (CWE-476) will occur in the call to strcpy().

Note that this example is also vulnerable to a buffer overflow (see CWE-119).

Observed Examples

Reference	Description
CVE-2007-3798	Unchecked return value leads to resultant integer overflow and code execution.
CVE-2006-4447	Program does not check return value when invoking functions to drop privileges, which could leave users with higher privileges than expected by forcing those functions to fail.
CVE-2006-2916	Program does not check return value when invoking functions to drop privileges, which could leave users with higher privileges than expected by forcing those functions to fail.

Potential Mitigations

Phase: Implementation

Check the results of all functions that return a value and verify that the value is expected.

Effectiveness: High

Checking the return value of the function will typically be sufficient, however beware of race conditions (CWE-362) in a concurrent environment.

Phase: Implementation

Ensure that you account for all possible return values from the function.

Phase: Implementation

When designing a function, make sure you return a value or throw an exception in case of an error.

Background Details

Many functions will return some value about the success of their actions. This will alert the program whether or not to handle any errors caused by that function.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to	Named Chain(s) this relationship pertains to
ChildOf	Weakness Class	227	Failure to Fulfill API Contract ('API Abuse')	**Development Concepts (primary)699** **Seven Pernicious Kingdoms (primary)700**	
ChildOf	Category	389	Error Conditions, Return Values, Status Codes	Development Concepts699	
ChildOf	Category	728	OWASP Top Ten 2004 Category A7 - Improper Error	**Weaknesses in OWASP Top Ten (2004)**	

			Handling	**(primary)711**	
ChildOf	Category	742	CERT C Secure Coding Section 08 - Memory Management (MEM)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**	
ChildOf	Weakness Class	754	Improper Check for Unusual or Exceptional Conditions	**Research Concepts (primary)1000**	
CanPrecede	Weakness Base	476	NULL Pointer Dereference	Research Concepts1000	Unchecked Return Value to NULL Pointer Dereference690
StartsChain	Compound Element: Chain	690	Unchecked Return Value to NULL Pointer Dereference	Named Chains709	Unchecked Return Value to NULL Pointer Dereference690
PeerOf	Weakness Base	273	Improper Check for Dropped Privileges	Research Concepts1000	

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Unchecked Return Value
CLASP			Ignored function return value
OWASP Top Ten 2004	A7	CWE More Specific	Improper Error Handling
CERT C Secure Coding	MEM32-C		Detect and handle memory allocation errors

References

[REF-7] Mark Dowd, John McDonald and Justin Schuh. "The Art of Software Security Assessment". Chapter 7, "Program Building Blocks" Page 341.. 1st Edition. Addison Wesley. 2006.

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 20, "Checking Returns" Page 624. 2nd Edition. Microsoft. 2002.

Content History

Submissions

Submission Date	Submitter	Organization	Source
	7 Pernicious Kingdoms		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-09-08	CWE Content Team	MITRE	Internal
	updated Common Consequences, Relationships, Other Notes, Taxonomy Mappings		
2008-11-24	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2009-01-12	CWE Content Team	MITRE	Internal
	updated Background Details, Demonstrative Examples, Description, Observed Examples, Other Notes, Potential Mitigations		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Relationships		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		
2009-07-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		
2009-12-28	CWE Content Team	MITRE	Internal
	updated Common Consequences, Demonstrative Examples, References		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples, Potential Mitigations, References		
2010-04-05	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		

BACK TO TOP

Improper Validation of Array Index

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

Description

Description Summary

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

Alternate Terms

out-of-bounds array index

index-out-of-range

array index underflow

Time of Introduction

- Implementation

Applicable Platforms

Languages

C: *(Often)*

C++: *(Often)*

Language-independent

Common Consequences

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

Likelihood of Exploit

High

Detection Methods

Automated Static Analysis

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

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

Effectiveness: High

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

Automated Dynamic Analysis

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

Black Box

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

Demonstrative Examples

Example 1

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

(Bad Code)

Example Language: **C**

```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**

```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

Incorrect Permission Assignment for Critical Resource

Weakness ID: 732 *(Weakness Class)* **Status:** Draft

Description

Description Summary

The software specifies permissions for a security-critical resource in a way that allows that resource to be read or modified by unintended actors.

Extended Description

When a resource is given a permissions setting that provides access to a wider range of actors than required, it could lead to the disclosure of sensitive information, or the modification of that resource by unintended parties. This is especially dangerous when the resource is related to program configuration, execution or sensitive user data.

Time of Introduction

- Architecture and Design
- Implementation
- Installation
- Operation

Applicable Platforms

Languages

Language-independent

Modes of Introduction

The developer may set loose permissions in order to minimize problems when the user first runs the program, then create documentation stating that permissions should be tightened. Since system administrators and users do not always read the documentation, this can result in insecure permissions being left unchanged.

The developer might make certain assumptions about the environment in which the software runs - e.g., that the software is running on a single-user system, or the software is only accessible to trusted administrators. When the software is running in a different environment, the permissions become a problem.

Common Consequences

Scope	Effect
Confidentiality	An attacker may be able to read sensitive information from the associated resource, such as credentials or configuration information stored in a file.
Integrity	An attacker may be able to modify critical properties of the associated resource to gain privileges, such as replacing a world-writable executable with a Trojan horse.
Availability	An attacker may be able to destroy or corrupt critical data in the associated resource, such as deletion of records from a database.

Likelihood of Exploit

Medium to High

Detection Methods

Automated Static Analysis

Automated static analysis may be effective in detecting permission problems for system resources such as files, directories, shared memory, device interfaces, etc. Automated techniques may be able to detect the use of library functions that modify permissions, then analyze function calls for arguments that contain potentially insecure values.

However, since the software's intended security policy might allow loose permissions for certain operations (such as publishing a file on a web server), automated static analysis may produce some false positives - i.e., warnings that do not have any security consequences or require any code changes.

When custom permissions models are used - such as defining who can read messages in a particular forum in a bulletin board system - these can be difficult to detect using automated static analysis. It may be possible to define custom signatures that identify any custom functions that implement the permission checks and assignments.

Automated Dynamic Analysis

Automated dynamic analysis may be effective in detecting permission problems for system resources such as files, directories, shared memory, device interfaces, etc.

However, since the software's intended security policy might allow loose permissions for certain operations (such as publishing a file on a web server), automated dynamic analysis may produce some false positives - i.e., warnings that do not have any security consequences or require any code changes.

When custom permissions models are used - such as defining who can read messages in a particular forum in a bulletin board system - these can be difficult to detect using automated dynamic analysis. It may be possible to define custom signatures that identify any custom functions that implement the permission checks and assignments.

Manual Static Analysis

Manual static analysis may be effective in detecting the use of custom permissions models and functions. The code could then be examined to identifying usage of the related functions. Then the human analyst could evaluate permission assignments in the context of the intended security model of the software.

Manual Dynamic Analysis

Manual dynamic analysis may be effective in detecting the use of custom permissions models and functions. The program could then be executed with a focus on exercising code paths that are related to the custom permissions. Then the human analyst could evaluate permission assignments in the context of the intended security model of the software.

Fuzzing

Fuzzing is not effective in detecting this weakness.

Demonstrative Examples

Example 1

The following code sets the umask of the process to 0 before creating a file and writing "Hello world" into the file.

(Bad Code)

Example Language: **C**

```
#define OUTFILE "hello.out"

umask(0);
FILE *out;
/* Ignore CWE-59 (link following) for brevity */
out = fopen(OUTFILE, "w");
if (out) {
fprintf(out, "hello world!\n");
fclose(out);
}
```

After running this program on a UNIX system, running the "ls -l" command might return the following output:

(Result)

```
-rw-rw-rw- 1 username 13 Nov 24 17:58 hello.out
```

The "rw-rw-rw-" string indicates that the owner, group, and world (all users) can read the file and write to it.

Example 2

The following code snippet might be used as a monitor to periodically record whether a web site is alive. To ensure that the file can always be modified, the code uses chmod() to make the file world-writable.

(Bad Code)

Example Language: **Perl**

```
$fileName = "secretFile.out";

if (-e $fileName) {
chmod 0777, $fileName;
}

my $outFH;
if (! open($outFH, ">>$fileName")) {
```

```
ExitError("Couldn't append to $fileName: $!");
}
my $dateString = FormatCurrentTime();
my $status = IsHostAlive("cwe.mitre.org");
print $outFH "$dateString cwe status: $status!\n";
close($outFH);
```

The first time the program runs, it might create a new file that inherits the permissions from its environment. A file listing might look like:

(Result)

```
-rw-r--r-- 1 username 13 Nov 24 17:58 secretFile.out
```

This listing might occur when the user has a default umask of 022, which is a common setting. Depending on the nature of the file, the user might not have intended to make it readable by everyone on the system.

The next time the program runs, however - and all subsequent executions - the chmod will set the file's permissions so that the owner, group, and world (all users) can read the file and write to it:

(Result)

```
-rw-rw-rw- 1 username 13 Nov 24 17:58 secretFile.out
```

Perhaps the programmer tried to do this because a different process uses different permissions that might prevent the file from being updated.

Example 3

The following command recursively sets world-readable permissions for a directory and all of its children:

(Bad Code)

Example Language: **Shell**

```
chmod -R ugo+r DIRNAME
```

If this command is run from a program, the person calling the program might not expect that all the files under the directory will be world-readable. If the directory is expected to contain private data, this could become a security problem.

Observed Examples

Reference	Description
CVE-2009-3482	Anti-virus product sets insecure "Everyone: Full Control" permissions for files under the "Program Files" folder, allowing attackers to replace executables with Trojan horses.
CVE-2009-3897	Product creates directories with 0777 permissions at installation, allowing users to gain privileges and access a socket used for authentication.
CVE-2009-3489	Photo editor installs a service with an insecure security descriptor, allowing users to stop or start the service, or execute commands as SYSTEM.
CVE-2009-3289	Library function copies a file to a new target and uses the source file's permissions for the target, which is incorrect when the source file is a symbolic link, which typically has 0777 permissions.
CVE-2009-0115	Device driver uses world-writable permissions for a socket file, allowing attackers to inject arbitrary commands.
CVE-2009-1073	LDAP server stores a cleartext password in a world-readable file.
CVE-2009-0141	Terminal emulator creates TTY devices with world-writable permissions, allowing an attacker to write to the terminals of other users.
CVE-2008-0662	VPN product stores user credentials in a registry key with "Everyone: Full Control" permissions, allowing attackers to steal the credentials.

CVE-2008-0322	Driver installs its device interface with "Everyone: Write" permissions.
CVE-2009-3939	Driver installs a file with world-writable permissions.
CVE-2009-3611	Product changes permissions to 0777 before deleting a backup; the permissions stay insecure for subsequent backups.
CVE-2007-6033	Product creates a share with "Everyone: Full Control" permissions, allowing arbitrary program execution.
CVE-2007-5544	Product uses "Everyone: Full Control" permissions for memory-mapped files (shared memory) in inter-process communication, allowing attackers to tamper with a session.
CVE-2005-4868	Database product uses read/write permissions for everyone for its shared memory, allowing theft of credentials.
CVE-2004-1714	Security product uses "Everyone: Full Control" permissions for its configuration files.
CVE-2001-0006	"Everyone: Full Control" permissions assigned to a mutex allows users to disable network connectivity.
CVE-2002-0969	Chain: database product contains buffer overflow that is only reachable through a .ini configuration file - which has "Everyone: Full Control" permissions.

Potential Mitigations

Phase: Implementation

When using a critical resource such as a configuration file, check to see if the resource has insecure permissions (such as being modifiable by any regular user), and generate an error or even exit the software if there is a possibility that the resource could have been modified by an unauthorized party.

Phase: Architecture and Design

Divide your application into anonymous, normal, privileged, and administrative areas. Reduce the attack surface by carefully defining distinct user groups, privileges, and/or roles. Map these against data, functionality, and the related resources. Then set the permissions accordingly. This will allow you to maintain more fine-grained control over your resources.

Phases: Implementation; Installation

During program startup, explicitly set the default permissions or umask to the most restrictive setting possible. Also set the appropriate permissions during program installation. This will prevent you from inheriting insecure permissions from any user who installs or runs the program.

Phase: System Configuration

For all configuration files, executables, and libraries, make sure that they are only readable and writable by the software's administrator.

Phase: Documentation

Do not suggest insecure configuration changes in your documentation, especially if those configurations can extend to resources and other software that are outside the scope of your own software.

Phase: Installation

Do not assume that the system administrator will manually change the configuration to the settings that you recommend in the manual.

Phase: Testing

Use tools and techniques that require manual (human) analysis, such as penetration testing, threat modeling, and interactive tools that allow the tester to record and modify an active session. These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules.

Phase: Testing

Use monitoring tools that examine the software's process as it interacts with the operating system and the network. This technique is useful in cases when source code is unavailable, if the software was not developed by you, or if you want to verify that the build phase did not introduce any new weaknesses. Examples include debuggers that directly attach to the running process; system-call tracing utilities such as truss (Solaris) and strace (Linux); system activity monitors such as FileMon, RegMon, Process Monitor, and other Sysinternals utilities (Windows); and sniffers and protocol analyzers that monitor network traffic.

Attach the monitor to the process and watch for library functions or system calls on OS resources such as files, directories, and shared memory. Examine the arguments to these calls to infer which permissions are being used.

Note that this technique is only useful for permissions issues related to system resources. It is not likely to detect application-level business rules that are related to permissions, such as if a user of a blog system marks a post as "private," but the blog system inadvertently marks it as "public."

Phases: Testing; System Configuration

Ensure that your software runs properly under the Federal Desktop Core Configuration (FDCC) or an equivalent hardening configuration guide, which many organizations use to limit the attack surface and potential risk of deployed software.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	275	Permission Issues	**Development Concepts (primary)699**
ChildOf	Weakness Class	668	Exposure of Resource to Wrong Sphere	**Research Concepts (primary)1000**
ChildOf	Category	753	2009 Top 25 - Porous Defenses	**Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)750**
ChildOf	Category	803	2010 Top 25 - Porous Defenses	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**
RequiredBy	Compound Element: Composite	689	Permission Race Condition During Resource Copy	Research Concepts1000
ParentOf	Weakness Variant	276	Incorrect Default Permissions	**Research Concepts (primary)1000**
ParentOf	Weakness Variant	277	Insecure Inherited Permissions	**Research Concepts (primary)1000**
ParentOf	Weakness Variant	278	Insecure Preserved Inherited Permissions	**Research Concepts (primary)1000**
ParentOf	Weakness Variant	279	Incorrect Execution-Assigned Permissions	**Research Concepts (primary)1000**
ParentOf	Weakness Base	281	Improper Preservation of Permissions	**Research Concepts (primary)1000**

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
232	Exploitation of Privilege/Trust	
1	Accessing Functionality Not Properly Constrained by ACLs	
17	Accessing, Modifying or Executing Executable Files	
60	Reusing Session IDs (aka Session Replay)	
61	Session Fixation	
62	Cross Site Request Forgery (aka Session Riding)	
122	Exploitation of Authorization	
180	Exploiting Incorrectly Configured Access Control Security Levels	
234	Hijacking a privileged process	

References

Mark Dowd, John McDonald and Justin Schuh. "The Art of Software Security Assessment". Chapter 9, "File Permissions." Page 495.. 1st Edition. Addison Wesley. 2006.

John Viega and Gary McGraw. "Building Secure Software". Chapter 8, "Access Control." Page 194.. 1st Edition. Addison-Wesley. 2002.

Maintenance Notes

The relationships between privileges, permissions, and actors (e.g. users and groups) need further refinement within the Research view. One complication is that these concepts apply to two different pillars, related to control of resources (CWE-664) and protection mechanism failures (CWE-396).

Content History

Submissions

Submission Date	Submitter	Organization	Source
2008-09-08			Internal CWE Team
	new weakness-focused entry for Research view.		

Modifications

Modification Date	Modifier	Organization	Source
2009-01-12	CWE Content Team	MITRE	Internal
	updated Description, Likelihood of Exploit, Name, Potential Mitigations, Relationships		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Potential Mitigations, Related Attack Patterns		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Name		
2009-12-28	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Demonstrative Examples, Detection Factors, Modes of Introduction, Observed Examples, Potential Mitigations, References		
2010-02-16	CWE Content Team	MITRE	Internal
	updated Relationships		
2010-04-05	CWE Content Team	MITRE	Internal
	updated Potential Mitigations, Related Attack Patterns		

Previous Entry Names

Change Date	Previous Entry Name
2009-01-12	Insecure Permission Assignment for Resource
2009-05-27	Insecure Permission Assignment for Critical Resource

BACK TO TOP

Time-of-check Time-of-use (TOCTOU) Race Condition

Weakness ID: 367 *(Weakness Base)* **Status:** Incomplete

Description

Description Summary

The software checks the state of a resource before using that resource, but the resource's state can change between the check and the use in a way that invalidates the results of the check. This can cause the software to perform invalid actions when the resource is in an unexpected state.

Extended Description

This weakness can be security-relevant when an attacker can influence the state of the resource between check and use. This can happen with shared resources such as files, memory, or even variables in multi-threaded programs.

Alternate Terms

TOCTTOU:	The TOCCTOU acronym expands to "Time Of Check To Time Of Use". Usage varies between TOCTOU and TOCTTOU.

Time of Introduction

- Implementation

Applicable Platforms

Languages

All

Common Consequences

Scope	Effect
Access Control	The attacker can gain access to otherwise unauthorized resources.
Access Control Authorization	Race conditions such as this kind may be employed to gain read or write access to resources which are not normally readable or writable by the user in question.
Integrity	The resource in question, or other resources (through the corrupted one), may be changed in undesirable ways by a malicious user.
Accountability	If a file or other resource is written in this method, as opposed to in a valid way, logging of the activity may not occur.
Non-Repudiation	In some cases it may be possible to delete files a malicious user might not otherwise have access to, such as log files.

Likelihood of Exploit

Low to Medium

Demonstrative Examples

Example 1

(Bad Code)

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

```
struct stat *sb;
...
lstat("...",sb); // it has not been updated since the last time it was read
printf("stated file\n");
if (sb->st_mtimespec==...){
print("Now updating things\n");
updateThings();
}
```

Potentially the file could have been updated between the time of the check and the lstat, especially since the printf has latency.

Example 2

The following code is from a program installed setuid root. The program performs certain file operations on behalf of non-privileged users, and uses access checks to ensure that it does not use its root privileges to perform operations that should otherwise be unavailable the current user. The program uses the access() system call to check if the person running the program has permission to access the specified file before it opens the file and performs the necessary operations.

(Bad Code)

Example Language: **C**

```
if(!access(file,W_OK)) {
f = fopen(file,"w+");
operate(f);
...
}
else {

fprintf(stderr,"Unable to open file %s.\n",file);
}
```

The call to access() behaves as expected, and returns 0 if the user running the program has the necessary permissions to write to the file, and -1 otherwise. However, because both access() and fopen() operate on filenames rather than on file handles, there is no guarantee that the file variable still refers to the same file on disk when it is passed to fopen() that it did when it was passed to access(). If an attacker replaces file after the call to access() with a symbolic link to a different file, the program will use its root privileges to operate on the file even if it is a file that the attacker would otherwise be unable to modify. By tricking the program into performing an operation that would otherwise be impermissible, the attacker has gained elevated privileges. This type of vulnerability is not limited to programs with root privileges. If the application is capable of performing any operation that the attacker would not otherwise be allowed perform, then it is a possible target.

Observed Examples

Reference	Description
CVE-2003-0813	
CVE-2004-0594	
CVE-2008-2958	chain: time-of-check time-of-use (TOCTOU) race condition in program allows bypass of protection mechanism that was designed to prevent symlink attacks.
CVE-2008-1570	chain: time-of-check time-of-use (TOCTOU) race condition in program allows bypass of protection mechanism that was designed to prevent symlink attacks.

Potential Mitigations

The most basic advice for TOCTOU vulnerabilities is to not perform a check before the use. This does not resolve the underlying issue of the execution of a function on a resource whose state and identity cannot be assured, but it does help to limit the false sense of security given by the check.

Phase: Implementation

When the file being altered is owned by the current user and group, set the effective gid and uid to that of the current user and group when executing this statement.

Do not rely on user-specified input to determine what path to format.

Phase: Architecture and Design

Limit the interleaving of operations on files from multiple processes.

Limit the spread of time (cycles) between the check and use of a resource.

Phase: Implementation

Recheck the resource after the use call to verify that the action was taken appropriately.

Phase: Architecture and Design

Ensure that some environmental locking mechanism can be used to protect resources effectively.

Phase: Implementation

Ensure that locking occurs before the check, as opposed to afterwards, such that the resource, as checked, is the same as it is when in use.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	361	Time and State	**Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Class	362	Race Condition	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ChildOf	Category	743	CERT C Secure Coding Section 09 - Input Output (FIO)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
PeerOf	Weakness Base	373	State Synchronization Error	Research Concepts1000
ParentOf	Weakness Base	363	Race Condition Enabling Link Following	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	609	Double-Checked Locking	**Research Concepts (primary)1000**
MemberOf	View	630	Weaknesses Examined by SAMATE	**Weaknesses Examined by SAMATE (primary)630**
PeerOf	Weakness Base	386	Symbolic Name not Mapping to Correct Object	Research Concepts1000

Relationship Notes

TOCTOU issues do not always involve symlinks, and not every symlink issue is a TOCTOU problem.

Research Gaps

Non-symlink TOCTOU issues are not reported frequently, but they are likely to occur in code that attempts to be secure.

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Time-of-check Time-of-use race condition
7 Pernicious Kingdoms			File Access Race Conditions: TOCTOU
CLASP			Time of check, time of use race condition
CERT C Secure Coding	FIO01-C		Be careful using functions that use file names for identification

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
27	Leveraging Race Conditions via Symbolic Links	
29	Leveraging Time-of-Check and Time-of-Use (TOCTOU) Race Conditions	

White Box Definitions

A weakness where code path has:

1. start statement that validates a system resource by name rather than by reference
2. end statement that accesses the system resource by the name

References

Dan Tsafrir, Tomer Hertz, David Wagner and Dilma Da Silva. "Portably Solving File TOCTTOU Races with Hardness Amplification". 2008-02-28. <http://www.usenix.org/events/fast08/tech/tsafrir.html>.

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		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Common Consequences, Relationships, Other Notes, Taxonomy Mappings		
2008-10-14	CWE Content Team	MITRE	Internal
	updated Description, Name, Relationships		
2008-11-24	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2009-01-12	CWE Content Team	MITRE	Internal
	updated Alternate Terms, Observed Examples, Other Notes, References, Relationship Notes, Relationships, Research Gaps		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		
2009-07-17	KDM Analytics		External
	Improved the White Box Definition		
2009-07-27	CWE Content Team	MITRE	Internal
	updated White Box Definitions		

Previous Entry Names

Change Date	Previous Entry Name
2008-10-14	Time-of-check Time-of-use Race Condition

BACK TO TOP

Information Leak Through Comments

Weakness ID: 615 *(Weakness Variant)* **Status:** Incomplete

Description

Description Summary

While adding general comments is very useful, some programmers tend to leave important data, such as: filenames related to the web application, old links or links which were not meant to be browsed by users, old code fragments, etc.

Extended Description

An attacker who finds these comments can map the application's structure and files, expose hidden parts of the site, and study the fragments of code to reverse engineer the application, which may help develop further attacks against the site.

Time of Introduction

- Implementation

Demonstrative Examples

Example 1

The following comment, embedded in a JSP, will be displayed in the resulting HTML output.

(Bad Code)

Example Languages: **HTML and JSP**

```
<!-- FIXME: calling this with more than 30 args kills the JDBC server -->
```

Observed Examples

Reference	Description
CVE-2007-6197	Version numbers and internal hostnames leaked in HTML comments.
CVE-2007-4072	CMS places full pathname of server in HTML comment.
CVE-2009-2431	blog software leaks real username in HTML comment.

Potential Mitigations

Remove comments which have sensitive information about the design/implementation of the application. Some of the comments may be exposed to the user and affect the security posture of the application.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Variant	540	Information Leak Through Source Code	**Development Concepts (primary)699** **Research Concepts (primary)1000**

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	Anonymous Tool Vendor (under NDA)		Externally Mined

Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-07-01	Sean Eidemiller	Cigital	External
	added/updated demonstrative examples		
2008-07-01	Eric Dalci	Cigital	External
	updated Potential Mitigations, Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2008-10-14	CWE Content Team	MITRE	Internal
	updated Description		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		

2009-07-27	CWE Content Team	MITRE	Internal
	updated Observed Examples, Taxonomy Mappings		

BACK TO TOP

Scanned Languages

Language	Hash Number	Change Date
CPP	0869982531058300	3/9/2017
JavaScript	0134728271507297	3/9/2017
VbScript	1337832853011295	3/9/2017