AndroidUSBCamera Scan Report

Project Name	AndroidUSBCamera
Scan Start	Saturday, June 22, 2024 12:29:26 AM
Preset	Checkmarx Default
Scan Time	00h:28m:35s
Lines Of Code Scanned	11331
Files Scanned	16
Report Creation Time	Saturday, June 22, 2024 12:59:26 AM
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081
Team	CxServer
Checkmarx Version	8.7.0
Scan Type	Full
Source Origin	LocalPath
Density	3/100 (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 v3.2	All
OWASP Top 10 2013	All
FISMA 2014	All
NIST SP 800-53	All
OWASP Top 10 2017	All
OWASP Mobile Top 10 2016	All

Excluded:

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

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

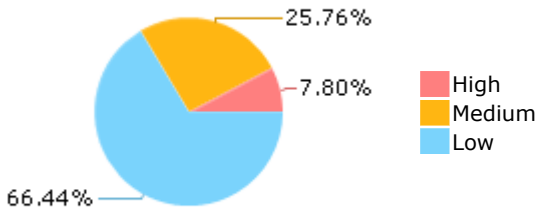
Results Limit

Results limit per query was set to 50

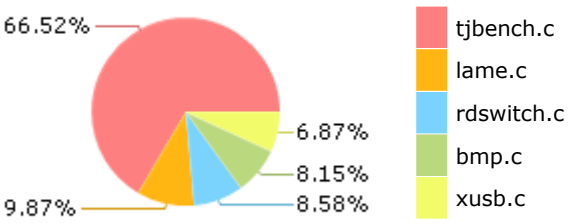
Selected Queries

Selected queries are listed in Result Summary

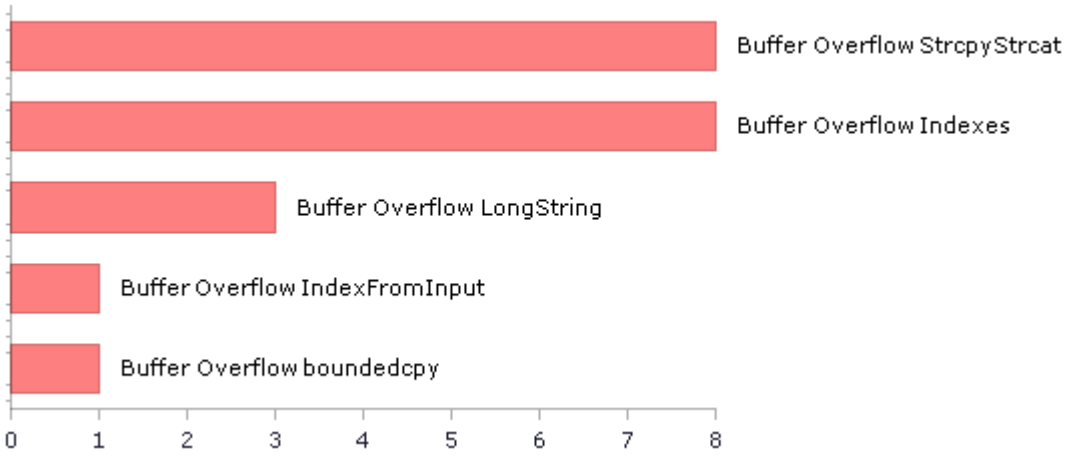
Result Summary



Most Vulnerable Files



Top 5 Vulnerabilities



Scan Summary - OWASP Top 10 2017

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

Category	Threat Agent	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection	App. Specific	EASY	COMMON	EASY	SEVERE	App. Specific	40	24
A2-Broken Authentication	App. Specific	EASY	COMMON	AVERAGE	SEVERE	App. Specific	37	37
A3-Sensitive Data Exposure	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App. Specific	0	0
A4-XML External Entities (XXE)	App. Specific	AVERAGE	COMMON	EASY	SEVERE	App. Specific	0	0
A5-Broken Access Control*	App. Specific	AVERAGE	COMMON	AVERAGE	SEVERE	App. Specific	3	3
A6-Security Misconfiguration	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A7-Cross-Site Scripting (XSS)	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A8-Insecure Deserialization	App. Specific	DIFFICULT	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A9-Using Components with Known Vulnerabilities*	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	MODERATE	App. Specific	27	27
A10-Insufficient Logging & Monitoring	App. Specific	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	App. Specific	0	0

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

Scan Summary - OWASP Top 10 2013

Further details and elaboration about vulnerabilities and risks can be found at: OWASP Top 10 2013

Category	Threat Agent	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	AVERAGE	SEVERE	ALL DATA	0	0
A2-Broken Authentication and Session Management	EXTERNAL, INTERNAL USERS	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	AFFECTED DATA AND FUNCTIONS	0	0
A3-Cross-Site Scripting (XSS)	EXTERNAL, INTERNAL, ADMIN USERS	AVERAGE	VERY WIDESPREAD	EASY	MODERATE	AFFECTED DATA AND SYSTEM	0	0
A4-Insecure Direct Object References	SYSTEM USERS	EASY	COMMON	EASY	MODERATE	EXPOSED DATA	3	3
A5-Security Misconfiguration	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	EASY	MODERATE	ALL DATA AND SYSTEM	0	0
A6-Sensitive Data Exposure	EXTERNAL, INTERNAL, ADMIN USERS, USERS BROWSERS	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	EXPOSED DATA	0	0
A7-Missing Function Level Access Control*	EXTERNAL, INTERNAL USERS	EASY	COMMON	AVERAGE	MODERATE	EXPOSED DATA AND FUNCTIONS	0	0
A8-Cross-Site Request Forgery (CSRF)	USERS BROWSERS	AVERAGE	COMMON	EASY	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A9-Using Components with Known Vulnerabilities*	EXTERNAL USERS, AUTOMATED TOOLS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	27	27
A10-Unvalidated Redirects and Forwards	USERS BROWSERS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0

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

Scan Summary - PCI DSS v3.2

Category	Issues Found	Best Fix Locations
PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection	3	3
PCI DSS (3.2) - 6.5.2 - Buffer overflows	42	26
PCI DSS (3.2) - 6.5.3 - Insecure cryptographic storage	0	0
PCI DSS (3.2) - 6.5.4 - Insecure communications	0	0
PCI DSS (3.2) - 6.5.5 - Improper error handling*	0	0
PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)	0	0
PCI DSS (3.2) - 6.5.8 - Improper access control	0	0
PCI DSS (3.2) - 6.5.9 - Cross-site request forgery	0	0
PCI DSS (3.2) - 6.5.10 - Broken authentication and session management	0	0

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

Scan Summary - FISMA 2014

Category	Description	Issues Found	Best Fix Locations
Access Control	Organizations must limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems) and to the types of transactions and functions that authorized users are permitted to exercise.	11	11
Audit And Accountability*	Organizations must: (i) create, protect, and retain information system audit records to the extent needed to enable the monitoring, analysis, investigation, and reporting of unlawful, unauthorized, or inappropriate information system activity; and (ii) ensure that the actions of individual information system users can be uniquely traced to those users so they can be held accountable for their actions.	3	3
Configuration Management	Organizations must: (i) establish and maintain baseline configurations and inventories of organizational information systems (including hardware, software, firmware, and documentation) throughout the respective system development life cycles; and (ii) establish and enforce security configuration settings for information technology products employed in organizational information systems.	97	10
Identification And Authentication*	Organizations must identify information system users, processes acting on behalf of users, or devices and authenticate (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing access to organizational information systems.	26	26
Media Protection	Organizations must: (i) protect information system media, both paper and digital; (ii) limit access to information on information system media to authorized users; and (iii) sanitize or destroy information system media before disposal or release for reuse.	0	0
System And Communications Protection	Organizations must: (i) monitor, control, and protect organizational communications (i.e., information transmitted or received by organizational information systems) at the external boundaries and key internal boundaries of the information systems; and (ii) employ architectural designs, software development techniques, and systems engineering principles that promote effective information security within organizational information systems.	0	0
System And Information Integrity	Organizations must: (i) identify, report, and correct information and information system flaws in a timely manner; (ii) provide protection from malicious code at appropriate locations within organizational information systems; and (iii) monitor information system security alerts and advisories and take appropriate actions in response.	6	6

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

Scan Summary - NIST SP 800-53

Category	Issues Found	Best Fix Locations
AC-12 Session Termination (P2)	0	0
AC-3 Access Enforcement (P1)	134	47
AC-4 Information Flow Enforcement (P1)	0	0
AC-6 Least Privilege (P1)	0	0
AU-9 Protection of Audit Information (P1)	0	0
CM-6 Configuration Settings (P2)	0	0
IA-5 Authenticator Management (P1)	0	0
IA-6 Authenticator Feedback (P2)	0	0
IA-8 Identification and Authentication (Non-Organizational Users) (P1)	0	0
SC-12 Cryptographic Key Establishment and Management (P1)	0	0
SC-13 Cryptographic Protection (P1)	0	0
SC-17 Public Key Infrastructure Certificates (P1)	0	0
SC-18 Mobile Code (P2)	0	0
SC-23 Session Authenticity (P1)*	0	0
SC-28 Protection of Information at Rest (P1)	0	0
SC-4 Information in Shared Resources (P1)	0	0
SC-5 Denial of Service Protection (P1)*	13	13
SC-8 Transmission Confidentiality and Integrity (P1)	0	0
SI-10 Information Input Validation (P1)*	31	15
SI-11 Error Handling (P2)*	34	34
SI-15 Information Output Filtering (P0)	0	0
SI-16 Memory Protection (P1)	5	5

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

Scan Summary - OWASP Mobile Top 10 2016

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

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

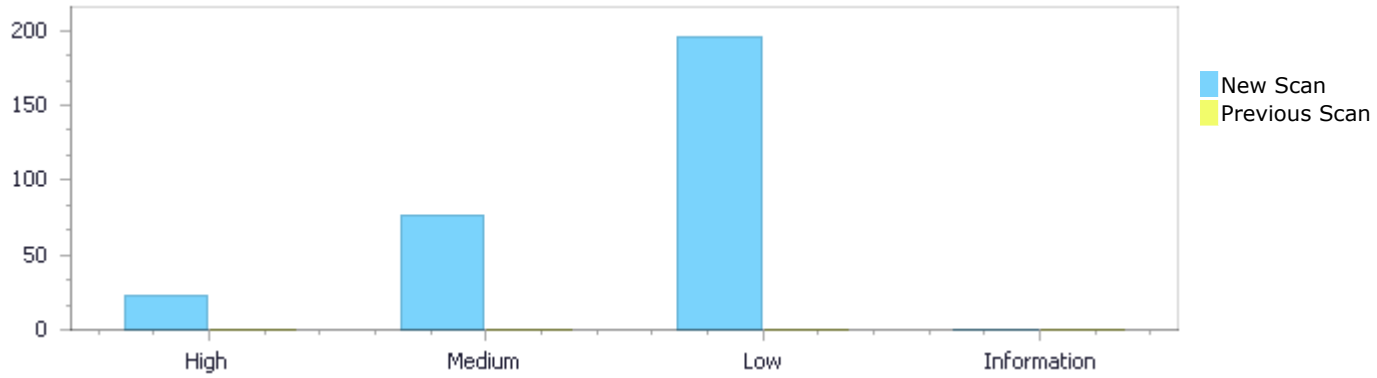
Scan Summary - Custom

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

Results Distribution By Status

First scan of the project

	High	Medium	Low	Information	Total
New Issues	23	76	196	0	295
Recurrent Issues	0	0	0	0	0
Total	23	76	196	0	295
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	23	76	196	0	295
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	23	76	196	0	295

Result Summary

Vulnerability Type	Occurrences	Severity
Buffer Overflow Indexes	8	High
Buffer Overflow StrcpyStrcat	8	High
Buffer Overflow LongString	3	High
Buffer Overflow boundedcpy	1	High
Buffer Overflow IndexFromInput	1	High

Buffer Overflow OutOfBound	1	High
Off by One Error in Arrays	1	High
Dangerous Functions	27	Medium
Divide By Zero	19	Medium
Buffer Overflow boundcpy WrongSizeParam	9	Medium
Memory Leak	6	Medium
Buffer Overflow AddressOfLocalVarReturned	4	Medium
Integer Overflow	4	Medium
Wrong Size t Allocation	4	Medium
Short Overflow	2	Medium
Double Free	1	Medium
Exposure of System Data to Unauthorized Control Sphere	97	Low
Unchecked Return Value	34	Low
Improper Resource Access Authorization	26	Low
Incorrect Permission Assignment For Critical Resources	11	Low
TOCTOU	11	Low
Use of Sizeof On a Pointer Type	4	Low
Arithmenic Operation On Boolean	3	Low
Potential Off by One Error in Loops	3	Low
Potential Path Traversal	3	Low
Unchecked Array Index	3	Low
Heuristic Buffer Overflow malloc	1	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
AndroidUSBCamera/tjbench.c	31
AndroidUSBCamera/lame.c	20
AndroidUSBCamera/ultrajsondec.c	9
AndroidUSBCamera/rdppm.c	9
AndroidUSBCamera/xusb.c	8
AndroidUSBCamera/json_writer.cpp	6
AndroidUSBCamera/ultrajsonenc.c	6
AndroidUSBCamera/rdswitch.c	5
AndroidUSBCamera/bmp.c	2
AndroidUSBCamera/misctest.cpp	2

Scan Results Details

Buffer Overflow Indexes

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

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description

Buffer Overflow Indexes\Path 1:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=65
Status	New

The size of the buffer used by get_text_gray_row in read_pbm_integer, at line 140 of AndroidUSBCamera/rdppm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pbm_getc passes to getc, at line 79 of AndroidUSBCamera/rdppm.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/rdppm.c	AndroidUSBCamera/rdppm.c
Line	85	152
Object	getc	read_pbm_integer

Code Snippet
File Name AndroidUSBCamera/rdppm.c
Method pbm_getc (FILE *infile)

```
....
85.     ch = getc(infile);
```

File Name AndroidUSBCamera/rdppm.c
Method get_text_gray_row (j_compress_ptr cinfo, cjpeg_source_ptr sinfo)

```
....
152.      *ptr++ = rescale[read_pbm_integer(cinfo, infile, maxval)];
```

Buffer Overflow Indexes\Path 2:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=66

Status	New

The size of the buffer used by get_text_rgb_row in read_pbm_integer, at line 159 of AndroidUSBCamera/rdppm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pbm_getc passes to getc, at line 79 of AndroidUSBCamera/rdppm.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/rdppm.c	AndroidUSBCamera/rdppm.c
Line	85	171
Object	getc	read_pbm_integer

Code Snippet
File Name AndroidUSBCamera/rdppm.c
Method pbm_getc (FILE *infile)

```
....
85.    ch = getc(infile);
```

File Name AndroidUSBCamera/rdppm.c

Method get_text_rgb_row (j_compress_ptr cinfo, cjpeg_source_ptr sinfo)

```
....
171.      *ptr++ = rescale[read_pbm_integer(cinfo, infile, maxval)];
```

Buffer Overflow Indexes\Path 3:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=67
Status	New

The size of the buffer used by get_text_rgb_row in read_pbm_integer, at line 159 of AndroidUSBCamera/rdppm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pbm_getc passes to getc, at line 79 of AndroidUSBCamera/rdppm.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/rdppm.c	AndroidUSBCamera/rdppm.c
Line	85	172
Object	getc	read_pbm_integer

Code Snippet
File Name AndroidUSBCamera/rdppm.c
Method pbm_getc (FILE *infile)

```
....
85.     ch = getc(infile);
```

File Name AndroidUSBCamera/rdppm.c

Method get_text_rgb_row (j_compress_ptr cinfo, cjpeg_source_ptr sinfo)

```
....
172.      *ptr++ = rescale[read_pbm_integer(cinfo, infile, maxval)];
```

Buffer Overflow Indexes\Path 4:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=68
Status	New

The size of the buffer used by get_text_rgb_row in read_pbm_integer, at line 159 of AndroidUSBCamera/rdppm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pbm_getc passes to getc, at line 79 of AndroidUSBCamera/rdppm.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/rdppm.c	AndroidUSBCamera/rdppm.c
Line	85	173
Object	getc	read_pbm_integer

Code Snippet

File Name AndroidUSBCamera/rdppm.c

Method pbm_getc (FILE *infile)

```
....
85.     ch = getc(infile);
```

File Name AndroidUSBCamera/rdppm.c

Method get_text_rgb_row (j_compress_ptr cinfo, cjpeg_source_ptr sinfo)

```
....
173.      *ptr++ = rescale[read_pbm_integer(cinfo, infile, maxval)];
```

Buffer Overflow Indexes\Path 5:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=69
Status	New

The size of the buffer used by get_text_gray_row in read_pbm_integer, at line 140 of AndroidUSBCamera/rdppm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pbm_getc passes to getc, at line 79 of AndroidUSBCamera/rdppm.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/rdppm.c	AndroidUSBCamera/rdppm.c
Line	88	152
Object	getc	read_pbm_integer

Code Snippet
File Name AndroidUSBCamera/rdppm.c
Method pbm_getc (FILE *infile)

```
....
88.         ch = getc(infile);
```

File Name AndroidUSBCamera/rdppm.c

Method get_text_gray_row (j_compress_ptr cinfo, cjpeg_source_ptr sinfo)

```
....
152.      *ptr++ = rescale[read_pbm_integer(cinfo, infile, maxval)];
```

Buffer Overflow Indexes\Path 6:

Severity High
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=70
Status New

The size of the buffer used by get_text_rgb_row in read_pbm_integer, at line 159 of AndroidUSBCamera/rdppm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pbm_getc passes to getc, at line 79 of AndroidUSBCamera/rdppm.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/rdppm.c	AndroidUSBCamera/rdppm.c
Line	88	171
Object	getc	read_pbm_integer

Code Snippet
File Name AndroidUSBCamera/rdppm.c
Method pbm_getc (FILE *infile)

```
....
88.         ch = getc(infile);
```

File Name AndroidUSBCamera/rdppm.c

Method get_text_rgb_row (j_compress_ptr cinfo, cjpeg_source_ptr sinfo)

```
....
171.      *ptr++ = rescale[read_pbm_integer(cinfo, infile, maxval)];
```

Buffer Overflow Indexes\Path 7:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=71
Status	New

The size of the buffer used by get_text_rgb_row in read_pbm_integer, at line 159 of AndroidUSBCamera/rdppm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pbm_getc passes to getc, at line 79 of AndroidUSBCamera/rdppm.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/rdppm.c	AndroidUSBCamera/rdppm.c
Line	88	172
Object	getc	read_pbm_integer

Code Snippet

File Name AndroidUSBCamera/rdppm.c

Method pbm_getc (FILE *infile)

```
....
88.         ch = getc(infile);
```

File Name AndroidUSBCamera/rdppm.c

Method get_text_rgb_row (j_compress_ptr cinfo, cjpeg_source_ptr sinfo)

```
....
172.      *ptr++ = rescale[read_pbm_integer(cinfo, infile, maxval)];
```

Buffer Overflow Indexes\Path 8:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=72
Status	New

The size of the buffer used by get_text_rgb_row in read_pbm_integer, at line 159 of AndroidUSBCamera/rdppm.c, is not properly verified before writing data to the buffer. This can enable a

buffer overflow attack, using the source buffer that pbm_getc passes to getc, at line 79 of AndroidUSBCamera/rdppm.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/rdppm.c	AndroidUSBCamera/rdppm.c
Line	88	173
Object	getc	read_pbm_integer

Code Snippet
File Name AndroidUSBCamera/rdppm.c
Method pbm_getc (FILE *infile)

```
....
88.         ch = getc(infile);
```

File Name AndroidUSBCamera/rdppm.c
Method get_text_rgb_row (j_compress_ptr cinfo, cjpeg_source_ptr sinfo)

```
....
173.      *ptr++ = rescale[read_pbm_integer(cinfo, infile, maxval)];
```

Buffer Overflow StrcpyStrcat

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

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description
Buffer Overflow StrcpyStrcat\Path 1:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=95
Status	New

The size of the buffer used by concatSep in dest, at line 1308 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that concatSep passes to dest, at line 1308 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1308	1311
Object	dest	dest

Code Snippet

File Name AndroidUSBCamera/lame.c
Method concatSep(char* dest, char const* sep, char const* str)

```
....
1308.  concatSep(char* dest, char const* sep, char const* str)
....
1311.      strcat(dest, str);
```

Buffer Overflow StrcpyStrcat\Path 2:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=96
Status	New

The size of the buffer used by concatSep in dest, at line 1308 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that concatSep passes to sep, at line 1308 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1308	1311
Object	sep	dest

Code Snippet
File Name AndroidUSBCamera/lame.c
Method concatSep(char* dest, char const* sep, char const* str)

```
....
1308.  concatSep(char* dest, char const* sep, char const* str)
....
1311.      strcat(dest, str);
```

Buffer Overflow StrcpyStrcat\Path 3:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=97
Status	New

The size of the buffer used by concatSep in dest, at line 1308 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that concatSep passes to str, at line 1308 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1308	1311
Object	str	dest

Code Snippet
File Name AndroidUSBCamera/lame.c
Method concatSep(char* dest, char const* sep, char const* str)

```
....
1308.  concatSep(char* dest, char const* sep, char const* str)
....
1311.      strcat(dest, str);
```

Buffer Overflow StrcpyStrcat\Path 4:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=98
Status	New

The size of the buffer used by concatSep in dest, at line 1308 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that concatSep passes to dest, at line 1308 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1308	1310
Object	dest	dest

Code Snippet
File Name AndroidUSBCamera/lame.c
Method concatSep(char* dest, char const* sep, char const* str)

```
....
1308.  concatSep(char* dest, char const* sep, char const* str)
....
1310.      if (*dest != 0) strcat(dest, sep);
```

Buffer Overflow StrcpyStrcat\Path 5:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=99
Status	New

The size of the buffer used by concatSep in dest, at line 1308 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that concatSep passes to sep, at line 1308 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1308	1310
Object	sep	dest

Code Snippet	
File Name	AndroidUSBCamera/lame.c
Method	concatSep(char* dest, char const* sep, char const* str)

```
....
1308.  concatSep(char* dest, char const* sep, char const* str)
....
1310.      if (*dest != 0) strcat(dest, sep);
```

Buffer Overflow StrcpyStrcat\Path 6:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=100
Status	New

The size of the buffer used by concatSep in dest, at line 1308 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that concatSep passes to str, at line 1308 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1308	1310
Object	str	dest

Code Snippet	
File Name	AndroidUSBCamera/lame.c
Method	concatSep(char* dest, char const* sep, char const* str)

```
....
1308.  concatSep(char* dest, char const* sep, char const* str)
....
1310.      if (*dest != 0) strcat(dest, sep);
```

Buffer Overflow StrcpyStrcat\Path 7:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=101
Status	New

The size of the buffer used by concatSep in sep, at line 1308 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that concatSep passes to sep, at line 1308 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1308	1310

Object	sep	sep

Code Snippet
File Name AndroidUSBCamera/lame.c
Method concatSep(char* dest, char const* sep, char const* str)

```
....
1308.  concatSep(char* dest, char const* sep, char const* str)
....
1310.      if (*dest != 0) strcat(dest, sep);
```

Buffer Overflow StrcpyStrcat\Path 8:

Severity High
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=102
Status New

The size of the buffer used by concatSep in str, at line 1308 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that concatSep passes to str, at line 1308 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1308	1311
Object	str	str

Code Snippet
File Name AndroidUSBCamera/lame.c
Method concatSep(char* dest, char const* sep, char const* str)

```
....
1308.  concatSep(char* dest, char const* sep, char const* str)
....
1311.      strcat(dest, str);
```

Buffer Overflow LongString

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

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description

Buffer Overflow LongString\Path 1:

Severity High
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=500

	81&pathid=1
Status	New

The size of the buffer used by decode_string in sur, at line 390 of AndroidUSBCamera/ultrajsondec.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that decode_string passes to "Unmatched "\"' when when decoding 'string'", at line 390 of AndroidUSBCamera/ultrajsondec.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	432	478
Object	"Unmatched ''\"' when when decoding 'string'"	sur

Code Snippet
File Name AndroidUSBCamera/ultrajsondec.c
Method FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
432.                    return SetError(ds, -1, "Unmatched ''\"' when
when decoding 'string'");
....
478.                                            sur[iSur] = (sur[iSur]
<< 4) + (JSUTF16) (*inputOffset - '0');
```

Buffer Overflow LongString\Path 2:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=2
Status	New

The size of the buffer used by decode_string in sur, at line 390 of AndroidUSBCamera/ultrajsondec.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that decode_string passes to "Unmatched "\"' when when decoding 'string'", at line 390 of AndroidUSBCamera/ultrajsondec.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	432	487
Object	"Unmatched ''\"' when when decoding 'string'"	sur

Code Snippet
File Name AndroidUSBCamera/ultrajsondec.c
Method FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
432.                    return SetError(ds, -1, "Unmatched ''\"' when
when decoding 'string'");
....
487.                                            sur[iSur] = (sur[iSur]
<< 4) + 10 + (JSUTF16) (*inputOffset - 'a');
```

Buffer Overflow LongString\Path 3:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=3
Status	New

The size of the buffer used by decode_string in sur, at line 390 of AndroidUSBCamera/ultrajsondec.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that decode_string passes to "Unmatched "\"' when when decoding 'string"', at line 390 of AndroidUSBCamera/ultrajsondec.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	432	496
Object	"Unmatched ''\"' when when decoding 'string"'	sur

Code Snippet
File Name AndroidUSBCamera/ultrajsondec.c
Method FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
432.                    return SetError(ds, -1, "Unmatched ''\"' when
when decoding 'string'");
....
496.                                            sur[iSur] = (sur[iSur]
<< 4) + 10 + (JSUTF16) (*inputOffset - 'A');
```

Buffer Overflow OutOfBound

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

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description

Buffer Overflow OutOfBound\Path 1:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=500

	81&pathid=4
Status	New

The size of the buffer used by test_device in r, at line 790 of AndroidUSBCamera/xusb.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that test_device passes to speed_name, at line 790 of AndroidUSBCamera/xusb.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	801	830
Object	speed_name	r

Code Snippet
File Name AndroidUSBCamera/xusb.c
Method static int test_device(uint16_t vid, uint16_t pid)

```
....
801.        const char* speed_name[5] = { "Unknown", "1.5 Mbit/s (USB
LowSpeed)", "12 Mbit/s (USB FullSpeed)",
....
830.              printf("              speed: %s\n", speed_name[r]);
```

Buffer Overflow boundedcpy

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

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description
Buffer Overflow boundedcpy\Path 1:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=89
Status	New

The size parameter srcsize in line 480 in file AndroidUSBCamera/tjbench.c is influenced by the user input argv in line 771 in file AndroidUSBCamera/tjbench.c. This may lead to a buffer overflow vulnerability, which may in turn result in malicious code execution.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	771	666
Object	argv	srcsize

Code Snippet
File Name AndroidUSBCamera/tjbench.c

Method	int main(int argc, char *argv[])

```
....
771.  int main(int argc, char *argv[])
```

File Name	AndroidUSBCamera/tjbench.c
Method	int decompTest(char *filename)

```
....
666.                    memcpy(jpegbuf[0], srcbuf, srcsize);
```

Off by One Error in Arrays

Query Path:
CPP\Cx\CPP Buffer Overflow\Off by One Error in Arrays Version:0

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
NIST SP 800-53: SI-16 Memory Protection (P1)
OWASP Top 10 2017: A1-Injection

Description
Off by One Error in Arrays\Path 1:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=90
Status	New

The buffer allocated by sizeof in AndroidUSBCamera/xusb.c at line 293 does not correctly account for the actual size of the value, resulting in an incorrect allocation that is off by one.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	301	301
Object	sizeof	sizeof

Code Snippet

File Name	AndroidUSBCamera/xusb.c
Method	static int set_xbox_actuators(libusb_device_handle *handle, uint8_t left, uint8_t right)

```
....
301.        output_report[1] = sizeof(output_report);
```

Buffer Overflow IndexFromInput

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

Categories

OWASP Top 10 2017: A1-Injection

Description

Buffer Overflow IndexFromInput\Path 1:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=295
Status	New

The size of the buffer used by test_device in r, at line 790 of AndroidUSBCamera/xusb.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that main passes to argv, at line 966 of AndroidUSBCamera/xusb.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	966	830
Object	argv	r

Code Snippet

File Name AndroidUSBCamera/xusb.c

Method int main(int argc, char** argv)

```
....
966.  int main(int argc, char** argv)
```

File Name AndroidUSBCamera/xusb.c

Method static int test_device(uint16_t vid, uint16_t pid)

```
....
830.              printf("              speed: %s\n", speed_name[r]);
```

Dangerous Functions

Query Path:

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

Categories

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

Description

Dangerous Functions\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=116
Status	New

The dangerous function, memcpy, was found in use at line 279 in AndroidUSBCamera/tjbench.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	328	328
Object	memcpy	memcpy

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
328.                    memcpy(&tmpbuf[pitch*i], &srcbuf[w*ps*i], w*ps);
```

Dangerous Functions\Path 2:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=117
Status New

The dangerous function, memcpy, was found in use at line 480 in AndroidUSBCamera/tjbench.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	666	666
Object	memcpy	memcpy

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
666.                    memcpy(jpegbuf[0], srcbuf, srcsize);
```

Dangerous Functions\Path 3:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=118
Status New

The dangerous function, memcpy, was found in use at line 390 in AndroidUSBCamera/ultrajsondec.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	416	416
Object	memcpy	memcpy

Code Snippet
File Name AndroidUSBCamera/ultrajsondec.c
Method FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
416.                    memcpy (ds->escStart, oldStart, escLen *
sizeof(wchar_t));
```

Dangerous Functions\Path 4:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=119
Status New

The dangerous function, memcpy, was found in use at line 94 in AndroidUSBCamera/ultrajsonenc.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/ultrajsonenc.c	AndroidUSBCamera/ultrajsonenc.c
Line	114	114
Object	memcpy	memcpy

Code Snippet
File Name AndroidUSBCamera/ultrajsonenc.c
Method void Buffer_Realloc (JSONObjectEncoder *enc, size_t cbNeeded)

```
....
114.              memcpy (enc->start, oldStart, offset);
```

Dangerous Functions\Path 5:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=120
Status New

The dangerous function, memcpy, was found in use at line 310 in AndroidUSBCamera/xusb.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	346	346
Object	memcpy	memcpy

Code Snippet
File Name AndroidUSBCamera/xusb.c
Method static int send_mass_storage_command(libusb_device_handle *handle, uint8_t endpoint, uint8_t lun,

```
....
346.        memcpy(cbw.CBWCB, cdb, cdb_len);
```

Dangerous Functions\Path 6:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=121
Status New

The dangerous function, sprintf, was found in use at line 768 in AndroidUSBCamera/misctest.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/misctest.cpp	AndroidUSBCamera/misctest.cpp
Line	773	773
Object	sprintf	sprintf

Code Snippet
File Name AndroidUSBCamera/misctest.cpp
Method void itoa_Writer_StringBufferVerify() {

```
....
773.          sprintf(buffer, "%d", randval[j]);
```

Dangerous Functions\Path 7:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=122
Status New

The dangerous function, sprintf, was found in use at line 781 in AndroidUSBCamera/misctest.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination

File	AndroidUSBCamera/misctest.cpp	AndroidUSBCamera/misctest.cpp
Line	785	785
Object	sprintf	sprintf

Code Snippet
File Name AndroidUSBCamera/misctest.cpp
Method void itoa_Writer_InsituStringStreamVerify() {

```
....
785.          sprintf(buffer, "%d", randval[j]);
```

Dangerous Functions\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=123
Status	New

The dangerous function, sprintf, was found in use at line 425 in AndroidUSBCamera/ultrajsonenc.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/ultrajsonenc.c	AndroidUSBCamera/ultrajsonenc.c
Line	493	493
Object	sprintf	sprintf

Code Snippet
File Name AndroidUSBCamera/ultrajsonenc.c
Method int Buffer_AppendDoubleUnchecked(JSOBJ obj, JSONObjectEncoder *enc, double value)

```
....
493.              enc->offset += sprintf(str, "%e", neg ? -value :
value);
```

Dangerous Functions\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=124
Status	New

The dangerous function, sprintf, was found in use at line 169 in AndroidUSBCamera/xusb.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination

File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	173	173
Object	sprintf	sprintf

Code Snippet
File Name AndroidUSBCamera/xusb.c
Method static char* uuid_to_string(const uint8_t* uuid)

```
....
173.        sprintf(uuid_string, "{%02x%02x%02x%02x-%02x%02x-%02x%02x-
%02x%02x-%02x%02x%02x%02x%02x%02x}",
```

Dangerous Functions\Path 10:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=125
Status New

The dangerous function, sscanf, was found in use at line 393 in AndroidUSBCamera/rdswitch.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	405	405
Object	sscanf	sscanf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method set_sample_factors (j_compress_ptr cinfo, char *arg)

```
....
405.        if (sscanf(arg, "%d%c%d%c", &val1, &ch1, &val2, &ch2) < 3)
```

Dangerous Functions\Path 11:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=126
Status New

The dangerous function, sscanf, was found in use at line 318 in AndroidUSBCamera/rdswitch.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c

Line	331	331
Object	sscanf	sscanf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method set_quality_ratings (j_compress_ptr cinfo, char *arg, boolean force_baseline)

```
....
331.        if (sscanf(arg, "%d%c", &val, &ch) < 1)
```

Dangerous Functions\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=127
Status	New

The dangerous function, sscanf, was found in use at line 358 in AndroidUSBCamera/rdswitch.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	371	371
Object	sscanf	sscanf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method set_quant_slots (j_compress_ptr cinfo, char *arg)

```
....
371.        if (sscanf(arg, "%d%c", &val, &ch) < 1)
```

Dangerous Functions\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=128
Status	New

The dangerous function, sscanf, was found in use at line 771 in AndroidUSBCamera/tjbench.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	801	801
Object	sscanf	sscanf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int main(int argc, char *argv[])

```
....
801.                    && sscanf(&temp[1], "%d", &maxqual)==1 &&
maxqual>minqual && maxqual>=1
```

Dangerous Functions\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=129
Status	New

The dangerous function, sscanf, was found in use at line 771 in AndroidUSBCamera/tjbench.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	842	842
Object	sscanf	sscanf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int main(int argc, char *argv[])

```
....
842.                            if(sscanf(argv[++i], "%d/%d", &temp1,
&temp2)==2)
```

Dangerous Functions\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=130
Status	New

The dangerous function, sscanf, was found in use at line 966 in AndroidUSBCamera/xusb.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	1061	1061
Object	sscanf	sscanf

Code Snippet	
File Name	AndroidUSBCamera/xusb.c
Method	int main(int argc, char** argv)

```
....
1061.                                   if (sscanf(argv[j], "%x:%x" ,
&tmp_vid, &tmp_pid) != 2) {
```

Dangerous Functions\Path 16:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=131
Status	New

The dangerous function, strcat, was found in use at line 1308 in AndroidUSBCamera/lame.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1310	1310
Object	strcat	strcat

Code Snippet	
File Name	AndroidUSBCamera/lame.c
Method	concatSep(char* dest, char const* sep, char const* str)

```
....
1310.       if (*dest != 0) strcat(dest, sep);
```

Dangerous Functions\Path 17:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=132
Status	New

The dangerous function, strcat, was found in use at line 1308 in AndroidUSBCamera/lame.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1311	1311
Object	strcat	strcat

Code Snippet	
File Name	AndroidUSBCamera/lame.c

Method	concatSep(char* dest, char const* sep, char const* str)

```
....
1311.       strcat(dest, str);
```

Dangerous Functions\Path 18:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=133
Status	New

The dangerous function, strlen, was found in use at line 771 in AndroidUSBCamera/tjbench.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	800	800
Object	strlen	strlen

Code Snippet	
File Name	AndroidUSBCamera/tjbench.c
Method	int main(int argc, char *argv[])

```
....
800.              if((temp=strchr(argv[2], '-'))!=NULL && strlen(temp)>1
```

Dangerous Functions\Path 19:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=134
Status	New

The dangerous function, strlen, was found in use at line 966 in AndroidUSBCamera/xusb.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	990	990
Object	strlen	strlen

Code Snippet	
File Name	AndroidUSBCamera/xusb.c
Method	int main(int argc, char** argv)

```
....
990.                    arglen = strlen(argv[j]);
```

Dangerous Functions\Path 20:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=135
Status	New

The dangerous function, strlen, was found in use at line 66 in AndroidUSBCamera/json_writer.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/json_writer.cpp	AndroidUSBCamera/json_writer.cpp
Line	74	74
Object	strlen	strlen

Code Snippet
File Name AndroidUSBCamera/json_writer.cpp
Method std::string valueToString(double value)

```
....
74.     char* ch = buffer + strlen(buffer) - 1;
```

Dangerous Functions\Path 21:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=136
Status	New

The dangerous function, strlen, was found in use at line 111 in AndroidUSBCamera/json_writer.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/json_writer.cpp	AndroidUSBCamera/json_writer.cpp
Line	119	119
Object	strlen	strlen

Code Snippet
File Name AndroidUSBCamera/json_writer.cpp
Method std::string valueToQuotedString(const char *value)

```
....
119.     unsigned maxsize = strlen(value)*2 + 3; //
allescaped+quotes+NULL
```

Dangerous Functions\Path 22:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=137
Status	New

The dangerous function, atoi, was found in use at line 771 in AndroidUSBCamera/tjbench.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	795	795
Object	atoi	atoi

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int main(int argc, char *argv[])

```
....
795.             if((minqual=atoi(argv[2]))<1 || minqual>100)
```

Dangerous Functions\Path 23:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=138
Status	New

The dangerous function, atoi, was found in use at line 771 in AndroidUSBCamera/tjbench.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	875	875
Object	atoi	atoi

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int main(int argc, char *argv[])

```
....
875.                        int temp=atoi(argv[++i]);
```

Dangerous Functions\Path 24:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=139
Status	New

The dangerous function, atoi, was found in use at line 771 in AndroidUSBCamera/tjbench.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	893	893
Object	atoi	atoi

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int main(int argc, char *argv[])

```
....
893.                        int temp=atoi(argv[++i]);
```

Dangerous Functions\Path 25:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=140
Status	New

The dangerous function, atoi, was found in use at line 771 in AndroidUSBCamera/tjbench.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	902	902
Object	atoi	atoi

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int main(int argc, char *argv[])

```
....
902.                                    int temp=atoi(argv[i]);
```

Dangerous Functions\Path 26:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=141
Status	New

The dangerous function, realloc, was found in use at line 390 in AndroidUSBCamera/ultrajsondec.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	409	409
Object	realloc	realloc

Code Snippet
File Name AndroidUSBCamera/ultrajsondec.c
Method FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
409.                    ds->escStart = (wchar_t *) ds->dec->realloc (ds->escStart, newSize * sizeof(wchar_t));
```

Dangerous Functions\Path 27:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=142
Status	New

The dangerous function, realloc, was found in use at line 94 in AndroidUSBCamera/ultrajsonenc.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	AndroidUSBCamera/ultrajsonenc.c	AndroidUSBCamera/ultrajsonenc.c
Line	107	107
Object	realloc	realloc

Code Snippet
File Name AndroidUSBCamera/ultrajsonenc.c
Method void Buffer_Realloc (JSONObjectEncoder *enc, size_t cbNeeded)

```
....
107.                enc->start = (char *) enc->realloc (enc->start,
newSize);
```

Divide By Zero

Query Path:
CPP\Cx\CPP Medium Threat\Divide By Zero Version:1

Description

Divide By Zero\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=46
Status	New

The application performs an illegal operation in start_input_ppm, in AndroidUSBCamera/rdppm.c. In line 307, the program attempts to divide by maxval, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input maxval in start_input_ppm of AndroidUSBCamera/rdppm.c, at line 307.

	Source	Destination
File	AndroidUSBCamera/rdppm.c	AndroidUSBCamera/rdppm.c
Line	434	434
Object	maxval	maxval

Code Snippet
File Name AndroidUSBCamera/rdppm.c
Method start_input_ppm (j_compress_ptr cinfo, cjpeg_source_ptr sinfo)

```
....
434.                                                maxval);
```

Divide By Zero\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=47
Status	New

The application performs an illegal operation in main, in AndroidUSBCamera/tjbench.c. In line 771, the program attempts to divide by temp2, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input temp2 in main of AndroidUSBCamera/tjbench.c, at line 771.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	846	846

Object	temp2	temp2

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int main(int argc, char *argv[])

```
....
846.                                    if((double)temp1/(double)temp2
```

Divide By Zero\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=48
Status	New

The application performs an illegal operation in decompTest, in AndroidUSBCamera/tjbench.c. In line 480, the program attempts to divide by tilew, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input tilew in decompTest of AndroidUSBCamera/tjbench.c, at line 480.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	536	536
Object	tilew	tilew

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
536.             ntilesw=(w+tilew-1)/tilew;  ntilesh=(h+tileh-1)/tileh;
```

Divide By Zero\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=49
Status	New

The application performs an illegal operation in decompTest, in AndroidUSBCamera/tjbench.c. In line 480, the program attempts to divide by tileh, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input tileh in decompTest of AndroidUSBCamera/tjbench.c, at line 480.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c

Line	536	536
Object	tileh	tileh

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
536.              ntilesw=(w+tilew-1)/tilew;   ntilesh=(h+tileh-1)/tileh;
```

Divide By Zero\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=50
Status	New

The application performs an illegal operation in decompTest, in AndroidUSBCamera/tjbench.c. In line 480, the program attempts to divide by _tilew, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input _tilew in decompTest of AndroidUSBCamera/tjbench.c, at line 480.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	593	593
Object	_tilew	_tilew

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
593.                    _ntilesw=(_w+_tilew-1)/_tilew;
```

Divide By Zero\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=51
Status	New

The application performs an illegal operation in decompTest, in AndroidUSBCamera/tjbench.c. In line 480, the program attempts to divide by _tileh, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input _tileh in decompTest of AndroidUSBCamera/tjbench.c, at line 480.

	Source	Destination

File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	594	594
Object	_tileh	_tileh

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
594.                    _ntilesh=(_h+_tileh-1)/_tileh;
```

Divide By Zero\Path 7:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=52
Status New

The application performs an illegal operation in decomp, in AndroidUSBCamera/tjbench.c. In line 104, the program attempts to divide by tilew, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input tilew in decomp of AndroidUSBCamera/tjbench.c, at line 104.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	116	116
Object	tilew	tilew

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
116.        int ntilesw=(w+tilew-1)/tilew, ntilesh=(h+tileh-1)/tileh;
```

Divide By Zero\Path 8:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=53
Status New

The application performs an illegal operation in decomp, in AndroidUSBCamera/tjbench.c. In line 104, the program attempts to divide by tileh, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input tileh in decomp of AndroidUSBCamera/tjbench.c, at line 104.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	116	116
Object	tileh	tileh

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
116.        int ntilesw=(w+tilew-1)/tilew, ntilesh=(h+tileh-1)/tileh;
```

Divide By Zero\Path 9:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=54
Status New

The application performs an illegal operation in decomp, in AndroidUSBCamera/tjbench.c. In line 104, the program attempts to divide by elapsed, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input elapsed in decomp of AndroidUSBCamera/tjbench.c, at line 104.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	194	194
Object	elapsed	elapsed

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
194.
      sigfig((double)(w*h)/1000000.*(double)iter/elapsed, 4, tempstr,
1024),
```

Divide By Zero\Path 10:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=55
Status New

The application performs an illegal operation in decomp, in AndroidUSBCamera/tjbench.c. In line 104, the program attempts to divide by elapsedDecode, which might be evaluate to 0 (zero) at time of division. This

value could be a hard-coded zero value, or received from external, untrusted input elapsedDecode in decomp of AndroidUSBCamera/tjbench.c, at line 104.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	198	198
Object	elapsedDecode	elapsedDecode

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
198.
      sigfig((double)(w*h)/1000000.*(double)iter/elapsedDecode, 4,
tempstr,
```

Divide By Zero\Path 11:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=56
Status New

The application performs an illegal operation in decomp, in AndroidUSBCamera/tjbench.c. In line 104, the program attempts to divide by elapsed, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input elapsed in decomp of AndroidUSBCamera/tjbench.c, at line 104.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	205	205
Object	elapsed	elapsed

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
205.                    doyuv? "Decomp to YUV":"Decompress   ",
(double)iter/elapsed);
```

Divide By Zero\Path 12:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=57

Status	New

The application performs an illegal operation in decomp, in AndroidUSBCamera/tjbench.c. In line 104, the program attempts to divide by elapsedDecode, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input elapsedDecode in decomp of AndroidUSBCamera/tjbench.c, at line 104.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	211	211
Object	elapsedDecode	elapsedDecode

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
211.                            (double)iter/elapsedDecode);
```

Divide By Zero\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=58
Status	New

The application performs an illegal operation in fullTest, in AndroidUSBCamera/tjbench.c. In line 279, the program attempts to divide by tilew, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input tilew in fullTest of AndroidUSBCamera/tjbench.c, at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	304	304
Object	tilew	tilew

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
304.              ntilesw=(w+tilew-1)/tilew;   ntilesh=(h+tileh-1)/tileh;
```

Divide By Zero\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=500

	81&pathid=59
Status	New

The application performs an illegal operation in fullTest, in AndroidUSBCamera/tjbench.c. In line 279, the program attempts to divide by elapsedEncode, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input elapsedEncode in fullTest of AndroidUSBCamera/tjbench.c, at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	392	392
Object	elapsedEncode	elapsedEncode

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
392.
      sigfig((double)(w*h)/1000000.*(double)iter/elapsedEncode, 4,
tempstr,
```

Divide By Zero\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=60
Status	New

The application performs an illegal operation in fullTest, in AndroidUSBCamera/tjbench.c. In line 279, the program attempts to divide by elapsed, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input elapsed in fullTest of AndroidUSBCamera/tjbench.c, at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	395	395
Object	elapsed	elapsed

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
395.
      sigfig((double)(w*h)/1000000.*(double)iter/elapsed, 4,
      tempstr, 1024),
```

Divide By Zero\Path 16:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=61
Status	New

The application performs an illegal operation in fullTest, in AndroidUSBCamera/tjbench.c. In line 279, the program attempts to divide by elapsedEncode, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input elapsedEncode in fullTest of AndroidUSBCamera/tjbench.c, at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	408	408
Object	elapsedEncode	elapsedEncode

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
408.                                    (double)iter/elapsedEncode);
```

Divide By Zero\Path 17:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=62
Status	New

The application performs an illegal operation in fullTest, in AndroidUSBCamera/tjbench.c. In line 279, the program attempts to divide by elapsed, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input elapsed in fullTest of AndroidUSBCamera/tjbench.c, at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	418	418
Object	elapsed	elapsed

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
418.                                doyuv? "Comp from YUV":"Compress     ",
(double)iter/elapsed);
```

Divide By Zero\Path 18:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=63
Status	New

The application performs an illegal operation in decompTest, in AndroidUSBCamera/tjbench.c. In line 480, the program attempts to divide by elapsed, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input elapsed in decompTest of AndroidUSBCamera/tjbench.c, at line 480.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	645	645
Object	elapsed	elapsed

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
645.
      sigfig((double)(w*h)/1000000./elapsed, 4, tempstr, 80),
```

Divide By Zero\Path 19:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=64
Status	New

The application performs an illegal operation in decompTest, in AndroidUSBCamera/tjbench.c. In line 480, the program attempts to divide by elapsed, which might be evaluate to 0 (zero) at time of division. This value could be a hard-coded zero value, or received from external, untrusted input elapsed in decompTest of AndroidUSBCamera/tjbench.c, at line 480.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	652	652
Object	elapsed	elapsed

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
652.                            printf("Transform     --> Frame rate:
%f fps\n", 1.0/elapsed);
```

Buffer Overflow boundcpy WrongSizeParam

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

Categories

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

Description

Buffer Overflow boundcpy WrongSizeParam\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=80
Status	New

The size of the buffer used by loadbmp in jpeg_compress_struct, at line 161 of AndroidUSBCamera/bmp.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that loadbmp passes to jpeg_compress_struct, at line 161 of AndroidUSBCamera/bmp.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	170	170
Object	jpeg_compress_struct	jpeg_compress_struct

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
170.        memset(&cinfo, 0, sizeof(struct jpeg_compress_struct));
```

Buffer Overflow boundcpy WrongSizeParam\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=81
Status	New

The size of the buffer used by savebmp in jpeg_decompress_struct, at line 244 of AndroidUSBCamera/bmp.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that savebmp passes to jpeg_decompress_struct, at line 244 of AndroidUSBCamera/bmp.c, to overwrite the target buffer.

	Source	Destination

File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	254	254
Object	jpeg_decompress_struct	jpeg_decompress_struct

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int savebmp(char *filename, unsigned char *buf, int w, int h, int srcpf,

```
....
254.        memset(&dinfo, 0, sizeof(struct jpeg_decompress_struct));
```

Buffer Overflow boundcpy WrongSizeParam\Path 3:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=82
Status New

The size of the buffer used by lame_init_bitstream in Namespace333292884, at line 2029 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that lame_init_bitstream passes to Namespace333292884, at line 2029 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	2041	2041
Object	Namespace333292884	Namespace333292884

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_init_bitstream(lame_global_flags * gfp)

```
....
2041.                    sizeof(gfc->ov_enc.bitrate_channelmode_hist));
```

Buffer Overflow boundcpy WrongSizeParam\Path 4:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=83
Status New

The size of the buffer used by lame_init_bitstream in Namespace333292884, at line 2029 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that lame_init_bitstream passes to Namespace333292884, at line 2029 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination

File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	2043	2043
Object	Namespace333292884	Namespace333292884

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_init_bitstream(lame_global_flags * gfp)

```
....
2043.                    sizeof(gfc->ov_enc.bitrate_blocktype_hist));
```

Buffer Overflow boundcpy WrongSizeParam\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=84
Status	New

The size of the buffer used by lame_init_old in lame_global_flags, at line 2303 of AndroidUSBCamera/lame.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that lame_init_old passes to lame_global_flags, at line 2303 of AndroidUSBCamera/lame.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	2310	2310
Object	lame_global_flags	lame_global_flags

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_init_old(lame_global_flags * gfp)

```
....
2310.      memset(gfp, 0, sizeof(lame_global_flags));
```

Buffer Overflow boundcpy WrongSizeParam\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=85
Status	New

The size of the buffer used by decode_string in escLen, at line 390 of AndroidUSBCamera/ultrajsondec.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that decode_string passes to escLen, at line 390 of AndroidUSBCamera/ultrajsondec.c, to overwrite the target buffer.

	Source	Destination

File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	416	416
Object	escLen	escLen

Code Snippet
File Name AndroidUSBCamera/ultrajsondec.c
Method FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
416.                    memcpy (ds->escStart, oldStart, escLen *
sizeof(wchar_t));
```

Buffer Overflow boundcpy WrongSizeParam\Path 7:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=86
Status New

The size of the buffer used by decode_string in long, at line 390 of AndroidUSBCamera/ultrajsondec.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that decode_string passes to long, at line 390 of AndroidUSBCamera/ultrajsondec.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	416	416
Object	long	long

Code Snippet
File Name AndroidUSBCamera/ultrajsondec.c
Method FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
416.                    memcpy (ds->escStart, oldStart, escLen *
sizeof(wchar_t));
```

Buffer Overflow boundcpy WrongSizeParam\Path 8:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=87
Status New

The size of the buffer used by decompTest in srcsize, at line 480 of AndroidUSBCamera/tjbench.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source

buffer that decompTest passes to srcsize, at line 480 of AndroidUSBCamera/tjbench.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	666	666
Object	srcsize	srcsize

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
666.                    memcpy(jpegbuf[0], srcbuf, srcsize);
```

Buffer Overflow boundcpy WrongSizeParam\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=88
Status	New

The size of the buffer used by send_mass_storage_command in cdb_len, at line 310 of AndroidUSBCamera/xusb.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that send_mass_storage_command passes to cdb_len, at line 310 of AndroidUSBCamera/xusb.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	346	346
Object	cdb_len	cdb_len

Code Snippet
File Name AndroidUSBCamera/xusb.c
Method static int send_mass_storage_command(libusb_device_handle *handle, uint8_t endpoint, uint8_t lun,

```
....
346.        memcpy(cbw.CBWCB, cdb, cdb_len);
```

Memory Leak

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

Categories

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

Description

Memory Leak\Path 1:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=144
Status	New

	Source	Destination
File	AndroidUSBCamera/DistributePipeline.cpp	AndroidUSBCamera/DistributePipeline.cpp
Line	91	91
Object	pipeline	pipeline

Code Snippet
File Name AndroidUSBCamera/DistributePipeline.cpp
Method static ID_TYPE nativeCreate(JNIEnv *env, jobject thiz) {

```
....
91.    DistributePipeline *pipeline = new DistributePipeline();
```

Memory Leak\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=145
Status	New

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	583	583
Object	ATH	ATH

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_init_params(lame_global_flags * gfp)

```
....
583.           gfc->ATH = calloc(1, sizeof(ATH_t));
```

Memory Leak\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=146
Status	New

	Source	Destination

File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	589	589
Object	rgdata	rgdata

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_init_params(lame_global_flags * gfp)

```
....
589.          gfc->sv_rpg.rgdata = calloc(1, sizeof(replaygain_t));
```

Memory Leak\Path 4:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=147
Status New

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	2314	2314
Object	internal_flags	internal_flags

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_init_old(lame_global_flags * gfp)

```
....
2314.      if (NULL == (gfc = gfp->internal_flags = calloc(1,
sizeof(lame_internal_flags))))
```

Memory Leak\Path 5:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=148
Status New

	Source	Destination
File	AndroidUSBCamera/ultrajsonenc.c	AndroidUSBCamera/ultrajsonenc.c
Line	811	811
Object	start	start

Code Snippet
File Name AndroidUSBCamera/ultrajsonenc.c

Method	char *JSON_EncodeObject(JSOBJ obj, JSONObjectEncoder *enc, char *_buffer, size_t _cbBuffer)

```
....
811.                enc->start = (char *) enc->malloc (_cbBuffer);
```

Memory Leak\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=149
Status	New

	Source	Destination
File	AndroidUSBCamera/ultrajsonenc.c	AndroidUSBCamera/ultrajsonenc.c
Line	113	113
Object	start	start

Code Snippet	
File Name	AndroidUSBCamera/ultrajsonenc.c
Method	void Buffer_Realloc (JSONObjectEncoder *enc, size_t cbNeeded)

```
....
113.                enc->start = (char *) enc->malloc (newSize);
```

Buffer Overflow AddressOfLocalVarReturned

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

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
NIST SP 800-53: SC-5 Denial of Service Protection (P1)
OWASP Top 10 2017: A1-Injection

Description

Buffer Overflow AddressOfLocalVarReturned\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=76
Status	New

The pointer buffer at AndroidUSBCamera/json_writer.cpp in line 66 is being used after it has been freed.

	Source	Destination
File	AndroidUSBCamera/json_writer.cpp	AndroidUSBCamera/json_writer.cpp
Line	75	75

Object	buffer	buffer

Code Snippet
File Name AndroidUSBCamera/json_writer.cpp
Method std::string valueToString(double value)

```
....
75.     if (*ch != '0') return buffer; // nothing to truncate, so save
time
```

Buffer Overflow AddressOfLocalVarReturned\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=77
Status	New

The pointer buffer at AndroidUSBCamera/json_writer.cpp in line 66 is being used after it has been freed.

	Source	Destination
File	AndroidUSBCamera/json_writer.cpp	AndroidUSBCamera/json_writer.cpp
Line	97	97
Object	buffer	buffer

Code Snippet
File Name AndroidUSBCamera/json_writer.cpp
Method std::string valueToString(double value)

```
....
97.         return buffer;
```

Buffer Overflow AddressOfLocalVarReturned\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=78
Status	New

The pointer buffer at AndroidUSBCamera/json_writer.cpp in line 66 is being used after it has been freed.

	Source	Destination
File	AndroidUSBCamera/json_writer.cpp	AndroidUSBCamera/json_writer.cpp
Line	99	99
Object	buffer	buffer

Code Snippet

File Name	AndroidUSBCamera/json_writer.cpp
Method	std::string valueToString(double value)

```
....
99.         return buffer;
```

Buffer Overflow AddressOfLocalVarReturned\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=79
Status	New

The pointer buffer at AndroidUSBCamera/json_writer.cpp in line 66 is being used after it has been freed.

	Source	Destination
File	AndroidUSBCamera/json_writer.cpp	AndroidUSBCamera/json_writer.cpp
Line	102	102
Object	buffer	buffer

Code Snippet

File Name	AndroidUSBCamera/json_writer.cpp
Method	std::string valueToString(double value)

```
....
102.     return buffer;
```

Wrong Size t Allocation

Query Path:
CPP\Cx\CPP Integer Overflow\Wrong Size t Allocation Version:0

Description

Wrong Size t Allocation\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=91
Status	New

The function srcsize in AndroidUSBCamera/tjbench.c at line 480 assigns an incorrectly calculated size to a buffer, resulting in a mismatch between the value being written and the size of the buffer it is being written into.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	497	497
Object	srcsize	srcsize

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
497.        if((srcbuf=(unsigned char *)malloc(srcsize))==NULL)
```

Wrong Size t Allocation\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=92
Status	New

The function _cbBuffer in AndroidUSBCamera/ultrajsonenc.c at line 788 assigns an incorrectly calculated size to a buffer, resulting in a mismatch between the value being written and the size of the buffer it is being written into.

	Source	Destination
File	AndroidUSBCamera/ultrajsonenc.c	AndroidUSBCamera/ultrajsonenc.c
Line	811	811
Object	_cbBuffer	_cbBuffer

Code Snippet
File Name AndroidUSBCamera/ultrajsonenc.c
Method char *JSON_EncodeObject(JSOBJ obj, JSONObjectEncoder *enc, char *_buffer, size_t _cbBuffer)

```
....
811.              enc->start = (char *) enc->malloc (_cbBuffer);
```

Wrong Size t Allocation\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=93
Status	New

The function newSize in AndroidUSBCamera/ultrajsondec.c at line 390 assigns an incorrectly calculated size to a buffer, resulting in a mismatch between the value being written and the size of the buffer it is being written into.

	Source	Destination
File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	415	415
Object	newSize	newSize

Code Snippet	
File Name	AndroidUSBCamera/ultrajsondec.c
Method	FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
415.                    ds->escStart = (wchar_t *) ds->dec->malloc
(newSize * sizeof(wchar_t));
```

Wrong Size t Allocation\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=94
Status	New

The function newSize in AndroidUSBCamera/ultrajsondec.c at line 390 assigns an incorrectly calculated size to a buffer, resulting in a mismatch between the value being written and the size of the buffer it is being written into.

	Source	Destination
File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	409	409
Object	newSize	newSize

Code Snippet	
File Name	AndroidUSBCamera/ultrajsondec.c
Method	FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
409.                    ds->escStart = (wchar_t *) ds->dec->realloc (ds-
>escStart, newSize * sizeof(wchar_t));
```

Integer Overflow

Query Path:
CPP\Cx\CPP Integer Overflow\Integer Overflow Version:0

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)

Description

Integer Overflow\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=104
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 2065 of AndroidUSBCamera/lame.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	2109	2109
Object	AssignExpr	AssignExpr

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_encode_flush(lame_global_flags * gfp, unsigned char *mp3buffer, int mp3buffer_size)

```
....
2109.          samples_to_encode += 16. / resample_ratio;
```

Integer Overflow\Path 2:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=105
Status New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 2065 of AndroidUSBCamera/lame.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	2121	2121
Object	AssignExpr	AssignExpr

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_encode_flush(lame_global_flags * gfp, unsigned char *mp3buffer, int mp3buffer_size)

```
....
2121.          bunch *= resample_ratio;
```

Integer Overflow\Path 3:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=106
Status New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 81 of AndroidUSBCamera/rdswitch.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	110	110
Object	AssignExpr	AssignExpr

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_quant_tables (j_compress_ptr cinfo, char *filename,

```
....
110.      table[0] = (unsigned int) val;
```

Integer Overflow\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=107
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 81 of AndroidUSBCamera/rdswitch.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	117	117
Object	AssignExpr	AssignExpr

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_quant_tables (j_compress_ptr cinfo, char *filename,

```
....
117.        table[i] = (unsigned int) val;
```

Short Overflow

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

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
FISMA 2014: System And Information Integrity
NIST SP 800-53: SI-10 Information Input Validation (P1)

Description
Short Overflow\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=108
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1809 of AndroidUSBCamera/lame.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1843	1843
Object	AssignExpr	AssignExpr

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_copy_inbuffer(lame_internal_flags* gfc,

```
....
1843.          COPY_AND_TRANSFORM(short int);
```

Short Overflow\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=109
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1809 of AndroidUSBCamera/lame.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	1843	1843
Object	AssignExpr	AssignExpr

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_copy_inbuffer(lame_internal_flags* gfc,

```
....
1843.          COPY_AND_TRANSFORM(short int);
```

Double Free

Query Path:
CPP\Cx\CPP Medium Threat\Double Free Version:1

Categories

NIST SP 800-53: SI-16 Memory Protection (P1)

Description

Double Free\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=143
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	273	474
Object	dstbuf	tmpbuf

Code Snippet

File Name AndroidUSBCamera/tjbench.c

Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
273.        if(dstbuf && dstbufalloc) free(dstbuf);
```

File Name AndroidUSBCamera/tjbench.c

Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
474.        if(tmpbuf) {free(tmpbuf);  tmpbuf=NULL;}
```

Exposure of System Data to Unauthorized Control Sphere

Query Path:

CPP\Cx\CPP Low Visibility\Exposure of System Data to Unauthorized Control Sphere Version:1

Categories

FISMA 2014: Configuration Management
NIST SP 800-53: AC-3 Access Enforcement (P1)

Description

Exposure of System Data to Unauthorized Control Sphere\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=187
Status	New

The system data read by decomp in the file AndroidUSBCamera/tjbench.c at line 104 is potentially exposed by decomp found in AndroidUSBCamera/tjbench.c at line 104.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	131	144
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
131.                    _throwunix("allocating destination buffer");
....
144.                    _throwunix("allocating YUV buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 2:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=188
Status New

The system data read by decomp in the file AndroidUSBCamera/tjbench.c at line 104 is potentially exposed by decomp found in AndroidUSBCamera/tjbench.c at line 104.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	144	144
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
144.                    _throwunix("allocating YUV buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 3:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=189
Status New

The system data read by decomp in the file AndroidUSBCamera/tjbench.c at line 104 is potentially exposed by decomp found in AndroidUSBCamera/tjbench.c at line 104.

	Source	Destination

File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	131	131
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
131.                    _throwunix("allocating destination buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 4:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=190
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	294	320
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
294.             _throwunix("allocating temporary image buffer");
....
320.                                 _throwunix("allocating JPEG tiles");
```

Exposure of System Data to Unauthorized Control Sphere\Path 5:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=191
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c

Line	308	320
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
308.                    _throwunix("allocating JPEG tile array");
....
320.                                 _throwunix("allocating JPEG tiles");
```

Exposure of System Data to Unauthorized Control Sphere\Path 6:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=192
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	312	320
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
312.                    _throwunix("allocating JPEG size array");
....
320.                                 _throwunix("allocating JPEG tiles");
```

Exposure of System Data to Unauthorized Control Sphere\Path 7:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=193
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination

File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	320	320
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
320.                                    _throwunix("allocating JPEG tiles");
```

Exposure of System Data to Unauthorized Control Sphere\Path 8:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=194
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	336	320
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
336.                            _throwunix("allocating YUV buffer");
....
320.                                    _throwunix("allocating JPEG tiles");
```

Exposure of System Data to Unauthorized Control Sphere\Path 9:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=195
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c

Line	433	320
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
433.                        _throwunix("opening reference image");
....
320.                              _throwunix("allocating JPEG tiles");
```

Exposure of System Data to Unauthorized Control Sphere\Path 10:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=196
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	435	320
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
435.                        _throwunix("writing reference image");
....
320.                              _throwunix("allocating JPEG tiles");
```

Exposure of System Data to Unauthorized Control Sphere\Path 11:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=197
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination

File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	294	435
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
294.              _throwunix("allocating temporary image buffer");
....
435.                           _throwunix("writing reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=198
Status	New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	308	435
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
308.                    _throwunix("allocating JPEG tile array");
....
435.                           _throwunix("writing reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=199
Status	New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	312	435
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
312.                    _throwunix("allocating JPEG size array");
....
435.                          _throwunix("writing reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 14:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=200
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	320	435
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
320.                          _throwunix("allocating JPEG tiles");
....
435.                        _throwunix("writing reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 15:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=201
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	336	435
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
336.                        _throwunix("allocating YUV buffer");
....
435.                        _throwunix("writing reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 16:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=202
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	433	435
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
433.                        _throwunix("opening reference image");
....
435.                        _throwunix("writing reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 17:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=203
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	435	435
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
435.                            _throwunix("writing reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 18:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=204
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	294	433
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
294.                _throwunix("allocating temporary image buffer");
....
433.                            _throwunix("opening reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 19:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=205
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination

File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	308	433
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
308.                    _throwunix("allocating JPEG tile array");
....
433.                          _throwunix("opening reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 20:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=206
Status	New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	312	433
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
312.                    _throwunix("allocating JPEG size array");
....
433.                          _throwunix("opening reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 21:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=207
Status	New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	320	433
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
320.                                    _throwunix("allocating JPEG tiles");
....
433.                                _throwunix("opening reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 22:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=208
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	336	433
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
336.                                _throwunix("allocating YUV buffer");
....
433.                                _throwunix("opening reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 23:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=209
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	433	433
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
433.                        _throwunix("opening reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 24:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=210
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	435	433
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
435.                        _throwunix("writing reference image");
....
433.                        _throwunix("opening reference image");
```

Exposure of System Data to Unauthorized Control Sphere\Path 25:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=211
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination

File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	294	336
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
294.              _throwunix("allocating temporary image buffer");
....
336.                           _throwunix("allocating YUV buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 26:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=212
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	308	336
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
308.                    _throwunix("allocating JPEG tile array");
....
336.                           _throwunix("allocating YUV buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 27:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=213
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	312	336
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
312.                    _throwunix("allocating JPEG size array");
....
336.                          _throwunix("allocating YUV buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 28:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=214
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	320	336
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
320.                               _throwunix("allocating JPEG tiles");
....
336.                          _throwunix("allocating YUV buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 29:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=215
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	336	336
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
336.                        _throwunix("allocating YUV buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 30:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=216
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	433	336
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
433.                        _throwunix("opening reference image");
....
336.                        _throwunix("allocating YUV buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 31:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=217
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination

File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	435	336
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
435.                        _throwunix("writing reference image");
....
336.                        _throwunix("allocating YUV buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 32:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=218
Status	New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	294	312
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
294.             _throwunix("allocating temporary image buffer");
....
312.                   _throwunix("allocating JPEG size array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 33:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=219
Status	New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	308	312
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
308.                    _throwunix("allocating JPEG tile array");
....
312.                    _throwunix("allocating JPEG size array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 34:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=220
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	312	312
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
312.                    _throwunix("allocating JPEG size array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 35:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=221
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination

File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	320	312
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
320.                                    _throwunix("allocating JPEG tiles");
....
312.                        _throwunix("allocating JPEG size array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 36:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=222
Status	New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	336	312
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
336.                              _throwunix("allocating YUV buffer");
....
312.                        _throwunix("allocating JPEG size array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 37:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=223
Status	New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	433	312
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
433.                        _throwunix("opening reference image");
....
312.                    _throwunix("allocating JPEG size array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 38:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=224
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	435	312
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
435.                        _throwunix("writing reference image");
....
312.                    _throwunix("allocating JPEG size array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 39:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=225
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	294	308
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
294.                _throwunix("allocating temporary image buffer");
....
308.                      _throwunix("allocating JPEG tile array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 40:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=226
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	308	308
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
308.                      _throwunix("allocating JPEG tile array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 41:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=227
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination

File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	312	308
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
312.                    _throwunix("allocating JPEG size array");
....
308.                    _throwunix("allocating JPEG tile array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 42:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=228
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	320	308
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
320.                                _throwunix("allocating JPEG tiles");
....
308.                    _throwunix("allocating JPEG tile array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 43:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=229
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	336	308
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
336.                        _throwunix("allocating YUV buffer");
....
308.                    _throwunix("allocating JPEG tile array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 44:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=230
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	433	308
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
433.                        _throwunix("opening reference image");
....
308.                    _throwunix("allocating JPEG tile array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 45:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=231
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	435	308
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
435.                            _throwunix("writing reference image");
....
308.                      _throwunix("allocating JPEG tile array");
```

Exposure of System Data to Unauthorized Control Sphere\Path 46:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=232
Status New

The system data read by fullTest in the file AndroidUSBCamera/tjbench.c at line 279 is potentially exposed by fullTest found in AndroidUSBCamera/tjbench.c at line 279.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	294	294
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
294.                _throwunix("allocating temporary image buffer");
```

Exposure of System Data to Unauthorized Control Sphere\Path 47:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=233
Status New

The system data read by decompTest in the file AndroidUSBCamera/tjbench.c at line 480 is potentially exposed by decompTest found in AndroidUSBCamera/tjbench.c at line 480.

	Source	Destination

File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	494	552
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
494.              _throwunix("opening file");
....
552.                                  _throwunix("allocating JPEG tiles");
```

Exposure of System Data to Unauthorized Control Sphere\Path 48:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=234
Status New

The system data read by decompTest in the file AndroidUSBCamera/tjbench.c at line 480 is potentially exposed by decompTest found in AndroidUSBCamera/tjbench.c at line 480.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	496	552
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
496.              _throwunix("determining file size");
....
552.                                  _throwunix("allocating JPEG tiles");
```

Exposure of System Data to Unauthorized Control Sphere\Path 49:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=235
Status New

The system data read by decompTest in the file AndroidUSBCamera/tjbench.c at line 480 is potentially exposed by decompTest found in AndroidUSBCamera/tjbench.c at line 480.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	498	552
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
498.              _throwunix("allocating memory");
....
552.                                  _throwunix("allocating JPEG tiles");
```

Exposure of System Data to Unauthorized Control Sphere\Path 50:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=236
Status New

The system data read by decompTest in the file AndroidUSBCamera/tjbench.c at line 480 is potentially exposed by decompTest found in AndroidUSBCamera/tjbench.c at line 480.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	500	552
Object	errno	printf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
500.              _throwunix("setting file position");
....
552.                                  _throwunix("allocating JPEG tiles");
```

Unchecked Return Value

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

Categories

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

Description

Unchecked Return Value\Path 1:

Severity Low

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=8
Status	New

The loadbmp method calls the snprintf function, at line 161 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	173	173
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
173.              _throw("loadbmp(): Invalid argument");
```

Unchecked Return Value\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=9
Status	New

The loadbmp method calls the snprintf function, at line 161 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	176	176
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
176.              _throwunix("loadbmp(): Cannot open input file");
```

Unchecked Return Value\Path 3:

Severity	Low
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=10
Status	New

The loadbmp method calls the snprintf function, at line 161 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	190	190
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
190.              _throwunix("loadbmp(): Could not read input file")
```

Unchecked Return Value\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=11
Status	New

The loadbmp method calls the snprintf function, at line 161 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	191	191
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
191.          else if(tempc==EOF) _throw("loadbmp(): Input file contains
no data");
```

Unchecked Return Value\Path 5:

Severity	Low
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=12
Status	New

The loadbmp method calls the snprintf function, at line 161 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	196	196
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
196.                    _throw("loadbmp(): Could not initialize bitmap
loader");
```

Unchecked Return Value\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=13
Status	New

The loadbmp method calls the snprintf function, at line 161 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	201	201
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
201.                    _throw("loadbmp(): Could not initialize bitmap
loader");
```

Unchecked Return Value\Path 7:

Severity	Low

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=14
Status	New

The loadbmp method calls the snprintf function, at line 161 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	203	203
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
203.        else _throw("loadbmp(): Unsupported file type");
```

Unchecked Return Value\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=15
Status	New

The loadbmp method calls the snprintf function, at line 161 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	217	217
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
217.              _throw("loadbmp(): Memory allocation failure");
```

Unchecked Return Value\Path 9:

Severity	Low
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=16
Status	New

The savebmp method calls the snprintf function, at line 244 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	257	257
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int savebmp(char *filename, unsigned char *buf, int w, int h, int srcpf,

```
....
257.                _throw("savebmp(): Invalid argument");
```

Unchecked Return Value\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=17
Status	New

The savebmp method calls the snprintf function, at line 244 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	260	260
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int savebmp(char *filename, unsigned char *buf, int w, int h, int srcpf,

```
....
260.                _throwunix("savebmp(): Cannot open output file");
```

Unchecked Return Value\Path 11:

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

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=18
Status	New

The savebmp method calls the snprintf function, at line 244 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	291	291
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int savebmp(char *filename, unsigned char *buf, int w, int h, int srcpf,

```
....
291.                    _throw("savebmp(): Could not initialize bitmap
writer");
```

Unchecked Return Value\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=19
Status	New

The savebmp method calls the snprintf function, at line 244 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	296	296
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int savebmp(char *filename, unsigned char *buf, int w, int h, int srcpf,

```
....
296.                    _throw("savebmp(): Could not initialize PPM
writer");
```

Unchecked Return Value\Path 13:

Severity	Low
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=20
Status	New

The itoa_Writer_StringBufferVerify method calls the sprintf function, at line 768 of AndroidUSBCamera/misctest.cpp. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/misctest.cpp	AndroidUSBCamera/misctest.cpp
Line	773	773
Object	sprintf	sprintf

Code Snippet
File Name AndroidUSBCamera/misctest.cpp
Method void itoa_Writer_StringBufferVerify() {

```
....
773.          sprintf(buffer, "%d", randval[j]);
```

Unchecked Return Value\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=21
Status	New

The itoa_Writer_InsituStringStreamVerify method calls the sprintf function, at line 781 of AndroidUSBCamera/misctest.cpp. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/misctest.cpp	AndroidUSBCamera/misctest.cpp
Line	785	785
Object	sprintf	sprintf

Code Snippet
File Name AndroidUSBCamera/misctest.cpp
Method void itoa_Writer_InsituStringStreamVerify() {

```
....
785.          sprintf(buffer, "%d", randval[j]);
```

Unchecked Return Value\Path 15:

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

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=22
Status	New

The *formatName method calls the snprintf function, at line 70 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	75	75
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method char *formatName(int subsamp, int cs, char *buf)

```
....
75.         snprintf(buf, 80, "%s %s", csName[cs],
subNameLong[subsamp]);
```

Unchecked Return Value\Path 16:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=23
Status	New

The *sigfig method calls the snprintf function, at line 82 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	86	86
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method char *sigfig(double val, int figs, char *buf, int len)

```
....
86.   if(digitsafterdecimal<1) snprintf(format, 80, "%%.0f");
```

Unchecked Return Value\Path 17:

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

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=24
Status	New

The *sigfig method calls the snprintf function, at line 82 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	87	87
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method char *sigfig(double val, int figs, char *buf, int len)

```
....
87.   else snprintf(format, 80, "%%.%df", digitsafterdecimal);
```

Unchecked Return Value\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=25
Status	New

The *sigfig method calls the snprintf function, at line 82 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	88	88
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method char *sigfig(double val, int figs, char *buf, int len)

```
....
88.   snprintf(buf, len, format, val);
```

Unchecked Return Value\Path 19:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=500

	81&pathid=26
Status	New

The decomp method calls the snprintf function, at line 104 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	121	121
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
121.              snprintf(qualstr, 6, "_Q%d", jpegqual);
```

Unchecked Return Value\Path 20:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=27
Status	New

The decomp method calls the snprintf function, at line 104 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	220	220
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
220.              snprintf(sizestr, 20, "%d_%d", sf.num, sf.denom);
```

Unchecked Return Value\Path 21:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=28

Status	New

The decomp method calls the snprintf function, at line 104 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	222	222
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
222.              snprintf(sizestr, 20, "%dx%d", tilew, tileh);
```

Unchecked Return Value\Path 22:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=29
Status	New

The decomp method calls the snprintf function, at line 104 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	223	223
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
223.        else snprintf(sizestr, 20, "full");
```

Unchecked Return Value\Path 23:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=30
Status	New

The decomp method calls the snprintf function, at line 104 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	225	225
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
225.              snprintf(tempstr, 1024, "%s_%s.%s", filename, sizestr,
ext);
```

Unchecked Return Value\Path 24:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=31
Status New

The decomp method calls the snprintf function, at line 104 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	227	227
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
227.              snprintf(tempstr, 1024, "%s_%s%s_%s.%s", filename,
subName[subsamp],
```

Unchecked Return Value\Path 25:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=32

Status	New

The decomp method calls the snprintf function, at line 104 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	234	234
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
234.        snprintf(ptr, 1024-(ptr-tempstr), "-err.%s", ext);
```

Unchecked Return Value\Path 26:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=33
Status	New

The fullTest method calls the snprintf function, at line 279 of AndroidUSBCamera/tjbench.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	430	430
Object	snprintf	snprintf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
430.                    snprintf(tempstr, 1024, "%s_%s_Q%d.jpg",
filename, subName[subsamp],
```

Unchecked Return Value\Path 27:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=34

Status	New

The uuid_to_string method calls the sprintf function, at line 169 of AndroidUSBCamera/xusb.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	173	173
Object	sprintf	sprintf

Code Snippet
File Name AndroidUSBCamera/xusb.c
Method static char* uuid_to_string(const uint8_t* uuid)

```
....
173.        sprintf(uuid_string, "{%02x%02x%02x%02x-%02x%02x-%02x%02x-
%02x%02x-%02x%02x%02x%02x%02x%02x}",
```

Unchecked Return Value\Path 28:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=35
Status	New

The valueToString method calls the sprintf_s function, at line 66 of AndroidUSBCamera/json_writer.cpp. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/json_writer.cpp	AndroidUSBCamera/json_writer.cpp
Line	70	70
Object	sprintf_s	sprintf_s

Code Snippet
File Name AndroidUSBCamera/json_writer.cpp
Method std::string valueToString(double value)

```
....
70.     sprintf_s(buffer, sizeof(buffer), "%#.16g", value);
```

Unchecked Return Value\Path 29:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=36

Status	New

The loadbmp method calls the Pointer function, at line 161 of AndroidUSBCamera/bmp.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	216	216
Object	Pointer	Pointer

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
216.        if((*buf=(unsigned char *)malloc((*w)*(*h)*dstps))==NULL)
```

Unchecked Return Value\Path 30:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=37
Status	New

The decode_string method calls the escStart function, at line 390 of AndroidUSBCamera/ultrajsondec.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	409	409
Object	escStart	escStart

Code Snippet
File Name AndroidUSBCamera/ultrajsondec.c
Method FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
409.                    ds->escStart = (wchar_t *) ds->dec->realloc (ds->escStart, newSize * sizeof(wchar_t));
```

Unchecked Return Value\Path 31:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=500

	81&pathid=38
Status	New

The decode_string method calls the escStart function, at line 390 of AndroidUSBCamera/ultrajsondec.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/ultrajsondec.c	AndroidUSBCamera/ultrajsondec.c
Line	415	415
Object	escStart	escStart

Code Snippet
File Name AndroidUSBCamera/ultrajsondec.c
Method FASTCALL_ATTR JSOBJ FASTCALL_MSVC decode_string (struct DecoderState *ds)

```
....
415.                    ds->escStart = (wchar_t *) ds->dec->malloc
(newSize * sizeof(wchar_t));
```

Unchecked Return Value\Path 32:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=39
Status	New

The *JSON_EncodeObject method calls the start function, at line 788 of AndroidUSBCamera/ultrajsonenc.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/ultrajsonenc.c	AndroidUSBCamera/ultrajsonenc.c
Line	811	811
Object	start	start

Code Snippet
File Name AndroidUSBCamera/ultrajsonenc.c
Method char *JSON_EncodeObject(JSOBJ obj, JSONObjectEncoder *enc, char *_buffer, size_t _cbBuffer)

```
....
811.              enc->start = (char *) enc->malloc (_cbBuffer);
```

Unchecked Return Value\Path 33:

Severity	Low
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=40
Status	New

The Buffer_Realloc method calls the start function, at line 94 of AndroidUSBCamera/ultrajsonenc.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/ultrajsonenc.c	AndroidUSBCamera/ultrajsonenc.c
Line	107	107
Object	start	start

Code Snippet
File Name AndroidUSBCamera/ultrajsonenc.c
Method void Buffer_Realloc (JSONObjectEncoder *enc, size_t cbNeeded)

```
....
107.                enc->start = (char *) enc->realloc (enc->start,
newSize);
```

Unchecked Return Value\Path 34:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=41
Status	New

The Buffer_Realloc method calls the start function, at line 94 of AndroidUSBCamera/ultrajsonenc.c. However, the code does not check the return value from this function, and thus would not detect runtime errors or other unexpected states.

	Source	Destination
File	AndroidUSBCamera/ultrajsonenc.c	AndroidUSBCamera/ultrajsonenc.c
Line	113	113
Object	start	start

Code Snippet
File Name AndroidUSBCamera/ultrajsonenc.c
Method void Buffer_Realloc (JSONObjectEncoder *enc, size_t cbNeeded)

```
....
113.                enc->start = (char *) enc->malloc (newSize);
```

Improper Resource Access Authorization

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

Categories

FISMA 2014: Identification And Authentication
NIST SP 800-53: AC-3 Access Enforcement (P1)
OWASP Top 10 2017: A2-Broken Authentication

Description

Improper Resource Access Authorization\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=150
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	501	501
Object	srcbuf	srcbuf

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
501.        if(fread(srcbuf, srcsize, 1, file)<1)
```

Improper Resource Access Authorization\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=151
Status	New

	Source	Destination
File	AndroidUSBCamera/example.c	AndroidUSBCamera/example.c
Line	307	307
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/example.c
Method read_JPEG_file (char *filename)

```
....
307.      fprintf(stderr, "can't open %s\n", filename);
```

Improper Resource Access Authorization\Path 3:

Severity	Low

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=152
Status	New

	Source	Destination
File	AndroidUSBCamera/example.c	AndroidUSBCamera/example.c
Line	115	115
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/example.c
Method write_JPEG_file (char *filename, int quality)

```
....
115.      fprintf(stderr, "can't open %s\n", filename);
```

Improper Resource Access Authorization\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=153
Status	New

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c
Line	481	481
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method main (int argc, char **argv)

```
....
481.      fprintf(stderr, "%s: only one input file\n", progname);
```

Improper Resource Access Authorization\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=154
Status	New

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c

Line	486	486
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method main (int argc, char **argv)

```
....
486.        fprintf(stderr, "%s: can't open %s\n", progname,
argv[argn]);
```

Improper Resource Access Authorization\Path 6:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=155
Status New

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c
Line	496	496
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method main (int argc, char **argv)

```
....
496.        fprintf(stderr, "%s: can't open stdin\n", progname);
```

Improper Resource Access Authorization\Path 7:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=156
Status New

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c
Line	156	156
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method next_marker (void)

```
....
156.      fprintf(stderr, "Warning: garbage data found in JPEG file\n");
```

Improper Resource Access Authorization\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=157
Status	New

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c
Line	407	407
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method usage (void)

```
....
407.    fprintf(stderr, "rdjpgcom displays any textual comments in a
JPEG file.\n");
```

Improper Resource Access Authorization\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=158
Status	New

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c
Line	409	409
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method usage (void)

```
....
409.    fprintf(stderr, "Usage: %s [switches] [inputfile]\n", progname);
```

Improper Resource Access Authorization\Path 10:

Severity	Low
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=159
Status	New

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c
Line	411	411
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method usage (void)

```
....
411.    fprintf(stderr, "Switches (names may be abbreviated):\n");
```

Improper Resource Access Authorization\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=160
Status	New

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c
Line	412	412
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method usage (void)

```
....
412.    fprintf(stderr, "  -raw        Display non-printable characters
in comments (unsafe)\n");
```

Improper Resource Access Authorization\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=161
Status	New

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c

Line	413	413
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method usage (void)

```
....
413.    fprintf(stderr, "  -verbose    Also display dimensions of JPEG
image\n");
```

Improper Resource Access Authorization\Path 13:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=162
Status New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	410	410
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method set_sample_factors (j_compress_ptr cinfo, char *arg)

```
....
410.          fprintf(stderr, "JPEG sampling factors must be 1..4\n");
```

Improper Resource Access Authorization\Path 14:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=163
Status New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	99	99
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_quant_tables (j_compress_ptr cinfo, char *filename,

```
....
99.      fprintf(stderr, "Can't open table file %s\n", filename);
```

Improper Resource Access Authorization\Path 15:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=164
Status	New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	106	106
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_quant_tables (j_compress_ptr cinfo, char *filename,

```
....
106.        fprintf(stderr, "Too many tables in file %s\n", filename);
```

Improper Resource Access Authorization\Path 16:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=165
Status	New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	113	113
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_quant_tables (j_compress_ptr cinfo, char *filename,

```
....
113.          fprintf(stderr, "Invalid table data in file %s\n",
filename);
```

Improper Resource Access Authorization\Path 17:

Severity	Low
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=166
Status	New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	130	130
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_quant_tables (j_compress_ptr cinfo, char *filename,

```
....
130.      fprintf(stderr, "Non-numeric data in file %s\n", filename);
```

Improper Resource Access Authorization\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=167
Status	New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	197	197
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_scan_script (j_compress_ptr cinfo, char *filename)

```
....
197.      fprintf(stderr, "Can't open scan definition file %s\n",
filename);
```

Improper Resource Access Authorization\Path 19:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=168
Status	New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c

Line	205	205
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_scan_script (j_compress_ptr cinfo, char *filename)

```
....
205.        fprintf(stderr, "Too many scans defined in file %s\n",
filename);
```

Improper Resource Access Authorization\Path 20:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=169
Status New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	213	213
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_scan_script (j_compress_ptr cinfo, char *filename)

```
....
213.          fprintf(stderr, "Too many components in one scan in file
%s\n",
```

Improper Resource Access Authorization\Path 21:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=170
Status New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	246	246
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_scan_script (j_compress_ptr cinfo, char *filename)

```
....
246.        fprintf(stderr, "Invalid scan entry format in file %s\n",
filename);
```

Improper Resource Access Authorization\Path 22:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=171
Status	New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	254	254
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_scan_script (j_compress_ptr cinfo, char *filename)

```
....
254.      fprintf(stderr, "Non-numeric data in file %s\n", filename);
```

Improper Resource Access Authorization\Path 23:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=172
Status	New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	376	376
Object	fprintf	fprintf

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method set_quant_slots (j_compress_ptr cinfo, char *arg)

```
....
376.          fprintf(stderr, "JPEG quantization tables are numbered
0..%d\n",
```

Improper Resource Access Authorization\Path 24:

Severity	Low

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=173
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	434	434
Object	fwrite	fwrite

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
434.                    if(fwrite(jpegbuf[0], jpegsize[0], 1, file)!=1)
```

Improper Resource Access Authorization\Path 25:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=174
Status	New

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	536	536
Object	fwrite	fwrite

Code Snippet
File Name AndroidUSBCamera/xusb.c
Method static int test_mass_storage(libusb_device_handle *handle, uint8_t endpoint_in, uint8_t endpoint_out)

```
....
536.                    if (fwrite(data, 1, (size_t)size, fd) !=
(unsigned int)size) {
```

Improper Resource Access Authorization\Path 26:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=175
Status	New

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	618	618
Object	fwrite	fwrite

Code Snippet
File Name AndroidUSBCamera/xusb.c
Method static int test_hid(libusb_device_handle *handle, uint8_t endpoint_in)

```
....
618.              if (fwrite(hid_report_descriptor, 1, descriptor_size,
fd) != descriptor_size) {
```

Incorrect Permission Assignment For Critical Resources

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

Categories

FISMA 2014: Access Control
NIST SP 800-53: AC-3 Access Enforcement (P1)
OWASP Top 10 2017: A2-Broken Authentication

Description

Incorrect Permission Assignment For Critical Resources\Path 1:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=176
Status New

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	175	175
Object	file	file

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
175.        if((file=fopen(filename, "rb"))==NULL)
```

Incorrect Permission Assignment For Critical Resources\Path 2:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=177

Status	New

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	259	259
Object	file	file

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int savebmp(char *filename, unsigned char *buf, int w, int h, int srcpf,

```
....
259.        if((file=fopen(filename, "wb"))==NULL)
```

Incorrect Permission Assignment For Critical Resources\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=178
Status	New

	Source	Destination
File	AndroidUSBCamera/example.c	AndroidUSBCamera/example.c
Line	306	306
Object	infile	infile

Code Snippet
File Name AndroidUSBCamera/example.c
Method read_JPEG_file (char *filename)

```
....
306.    if ((infile = fopen(filename, "rb")) == NULL) {
```

Incorrect Permission Assignment For Critical Resources\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=179
Status	New

	Source	Destination
File	AndroidUSBCamera/example.c	AndroidUSBCamera/example.c
Line	114	114
Object	outfile	outfile

Code Snippet
File Name AndroidUSBCamera/example.c
Method write_JPEG_file (char *filename, int quality)

```
....
114.    if ((outfile = fopen(filename, "wb")) == NULL) {
```

Incorrect Permission Assignment For Critical Resources\Path 5:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=180
Status New

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c
Line	485	485
Object	infile	infile

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method main (int argc, char **argv)

```
....
485.      if ((infile = fopen(argv[argn], READ_BINARY)) == NULL) {
```

Incorrect Permission Assignment For Critical Resources\Path 6:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=181
Status New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	98	98
Object	fp	fp

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_quant_tables (j_compress_ptr cinfo, char *filename,

```
....
98.    if ((fp = fopen(filename, "r")) == NULL) {
```

Incorrect Permission Assignment For Critical Resources\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=182
Status	New

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	196	196
Object	fp	fp

Code Snippet
File Name AndroidUSBCamera/rdswitch.c
Method read_scan_script (j_compress_ptr cinfo, char *filename)

```
....
196.    if ((fp = fopen(filename, "r")) == NULL) {
```

Incorrect Permission Assignment For Critical Resources\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=183
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	432	432
Object	file	file

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
432.                    if((file=fopen(tempstr, "wb"))==NULL)
```

Incorrect Permission Assignment For Critical Resources\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=184
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	493	493
Object	file	file

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
493.        if((file=fopen(filename, "rb"))==NULL)
```

Incorrect Permission Assignment For Critical Resources\Path 10:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=185
Status New

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	535	535
Object	fd	fd

Code Snippet
File Name AndroidUSBCamera/xusb.c
Method static int test_mass_storage(libusb_device_handle *handle, uint8_t endpoint_in, uint8_t endpoint_out)

```
....
535.              if ((binary_dump) && ((fd = fopen(binary_name, "w"))
!= NULL)) {
```

Incorrect Permission Assignment For Critical Resources\Path 11:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=186
Status New

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	617	617
Object	fd	fd

Code Snippet	
File Name	AndroidUSBCamera/xusb.c
Method	static int test_hid(libusb_device_handle *handle, uint8_t endpoint_in)

```
....
617.        if ((binary_dump) && ((fd = fopen(binary_name, "w")) !=
NULL)) {
```

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-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=284
Status	New

The loadbmp method in AndroidUSBCamera/bmp.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	175	175
Object	fopen	fopen

Code Snippet	
File Name	AndroidUSBCamera/bmp.c
Method	int loadbmp(char *filename, unsigned char **buf, int *w, int *h,

```
....
175.        if((file=fopen(filename, "rb"))==NULL)
```

TOCTOU\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=285
Status	New

The savebmp method in AndroidUSBCamera/bmp.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/bmp.c	AndroidUSBCamera/bmp.c
Line	259	259

Object	fopen	fopen

Code Snippet
File Name AndroidUSBCamera/bmp.c
Method int savebmp(char *filename, unsigned char *buf, int w, int h, int srcpf,

```
....
259.        if((file=fopen(filename, "wb"))==NULL)
```

TOCTOU\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=286
Status	New

The read_JPEG_file method in AndroidUSBCamera/example.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/example.c	AndroidUSBCamera/example.c
Line	306	306
Object	fopen	fopen

Code Snippet
File Name AndroidUSBCamera/example.c
Method read_JPEG_file (char *filename)

```
....
306.    if ((infile = fopen(filename, "rb")) == NULL) {
```

TOCTOU\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=287
Status	New

The write_JPEG_file method in AndroidUSBCamera/example.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/example.c	AndroidUSBCamera/example.c
Line	114	114

Object	fopen	fopen

Code Snippet
File Name AndroidUSBCamera/example.c
Method write_JPEG_file (char *filename, int quality)

```
....
114.    if ((outfile = fopen(filename, "wb")) == NULL) {
```

TOCTOU\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=288
Status	New

The main method in AndroidUSBCamera/rdjpgcom.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c
Line	485	485
Object	fopen	fopen

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method main (int argc, char **argv)

```
....
485.      if ((infile = fopen(argv[argn], READ_BINARY)) == NULL) {
```

TOCTOU\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=289
Status	New

The read_quant_tables method in AndroidUSBCamera/rdswitch.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	98	98
Object	fopen	fopen

Code Snippet	
File Name	AndroidUSBCamera/rdswitch.c
Method	read_quant_tables (j_compress_ptr cinfo, char *filename,

```
....
98.    if ((fp = fopen(filename, "r")) == NULL) {
```

TOCTOU\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=290
Status	New

The read_scan_script method in AndroidUSBCamera/rdswitch.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/rdswitch.c	AndroidUSBCamera/rdswitch.c
Line	196	196
Object	fopen	fopen

Code Snippet	
File Name	AndroidUSBCamera/rdswitch.c
Method	read_scan_script (j_compress_ptr cinfo, char *filename)

```
....
196.    if ((fp = fopen(filename, "r")) == NULL) {
```

TOCTOU\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=291
Status	New

The fullTest method in AndroidUSBCamera/tjbench.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	432	432
Object	fopen	fopen

Code Snippet

File Name	AndroidUSBCamera/tjbench.c
Method	int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
432.                    if((file=fopen(tempstr, "wb"))==NULL)
```

TOCTOU\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=292
Status	New

The decompTest method in AndroidUSBCamera/tjbench.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	493	493
Object	fopen	fopen

Code Snippet	
File Name	AndroidUSBCamera/tjbench.c
Method	int decompTest(char *filename)

```
....
493.        if((file=fopen(filename, "rb"))==NULL)
```

TOCTOU\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=293
Status	New

The test_mass_storage method in AndroidUSBCamera/xusb.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	535	535
Object	fopen	fopen

Code Snippet	
File Name	AndroidUSBCamera/xusb.c

Method	static int test_mass_storage(libusb_device_handle *handle, uint8_t endpoint_in, uint8_t endpoint_out)

```
....
535.              if ((binary_dump) && ((fd = fopen(binary_name, "w"))
!= NULL)) {
```

TOCTOU\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=294
Status	New

The test_hid method in AndroidUSBCamera/xusb.c file utilizes fopen that is accessed by other concurrent functionality in a way that is not thread-safe, which may result in a Race Condition over this resource.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c
Line	617	617
Object	fopen	fopen

Code Snippet	
File Name	AndroidUSBCamera/xusb.c
Method	static int test_hid(libusb_device_handle *handle, uint8_t endpoint_in)

```
....
617.          if ((binary_dump) && ((fd = fopen(binary_name, "w")) !=
NULL)) {
```

Use of Sizeof On a Pointer Type

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

Description

Use of Sizeof On a Pointer Type\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=42
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	306	306
Object	sizeof	sizeof

Code Snippet

File Name	AndroidUSBCamera/tjbench.c
Method	int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
306.                if((jpegbuf=(unsigned char **)malloc(sizeof(unsigned
char *)
```

Use of Sizeof On a Pointer Type\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=43
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	309	309
Object	sizeof	sizeof

Code Snippet

File Name	AndroidUSBCamera/tjbench.c
Method	int fullTest(unsigned char *srcbuf, int w, int h, int subsamp, int jpegqual,

```
....
309.                memset(jpegbuf, 0, sizeof(unsigned char
*)*ntilesw*ntilesh);
```

Use of Sizeof On a Pointer Type\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=44
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	538	538
Object	sizeof	sizeof

Code Snippet

File Name	AndroidUSBCamera/tjbench.c
Method	int decompTest(char *filename)

```
....
538.                if((jpegbuf=(unsigned char **)malloc(sizeof(unsigned
char *)
```

Use of Sizeof On a Pointer Type\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=45
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	541	541
Object	sizeof	sizeof

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
541.              memset(jpegbuf, 0, sizeof(unsigned char
*)*ntilesw*ntilesh);
```

Potential Path Traversal

Query Path:
CPP\Cx\CPP Low Visibility\Potential Path Traversal Version:0

Categories

OWASP Top 10 2013: A4-Insecure Direct Object References
OWASP Top 10 2017: A5-Broken Access Control

Description

Potential Path Traversal\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=5
Status	New

Method main at line 449 of AndroidUSBCamera/rdjpgcom.c gets user input from the argv element. This element's value then flows through the code and is eventually used in a file path for local disk access in main at line 449 of AndroidUSBCamera/rdjpgcom.c. This may cause a Path Traversal vulnerability.

	Source	Destination
File	AndroidUSBCamera/rdjpgcom.c	AndroidUSBCamera/rdjpgcom.c
Line	449	485
Object	argv	argv

Code Snippet
File Name AndroidUSBCamera/rdjpgcom.c
Method main (int argc, char **argv)

```
....
449.  main (int argc, char **argv)
....
485.      if ((infile = fopen(argv[argn], READ_BINARY)) == NULL) {
```

Potential Path Traversal\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=6
Status	New

Method main at line 771 of AndroidUSBCamera/tjbench.c gets user input from the argv element. This element's value then flows through the code and is eventually used in a file path for local disk access in decompTest at line 480 of AndroidUSBCamera/tjbench.c. This may cause a Path Traversal vulnerability.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	771	493
Object	argv	filename

Code Snippet

File Name AndroidUSBCamera/tjbench.c

Method int main(int argc, char *argv[])

```
....
771.  int main(int argc, char *argv[])
```

File Name AndroidUSBCamera/tjbench.c

Method int decompTest(char *filename)

```
....
493.        if((file=fopen(filename, "rb"))==NULL)
```

Potential Path Traversal\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=7
Status	New

Method main at line 966 of AndroidUSBCamera/xusb.c gets user input from the argv element. This element's value then flows through the code and is eventually used in a file path for local disk access in test_hid at line 602 of AndroidUSBCamera/xusb.c. This may cause a Path Traversal vulnerability.

	Source	Destination
File	AndroidUSBCamera/xusb.c	AndroidUSBCamera/xusb.c

Line	966	617
Object	argv	binary_name

Code Snippet
File Name AndroidUSBCamera/xusb.c
Method int main(int argc, char** argv)

```
....
966.  int main(int argc, char** argv)
```

File Name AndroidUSBCamera/xusb.c
Method static int test_hid(libusb_device_handle *handle, uint8_t endpoint_in)

```
....
617.        if ((binary_dump) && ((fd = fopen(binary_name, "w")) !=
NULL)) {
```

Potential Off by One Error in Loops

Query Path:
CPP\Cx\CPP Heuristic\Potential Off by One Error in Loops Version:1

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection
NIST SP 800-53: SI-16 Memory Protection (P1)
OWASP Top 10 2017: A1-Injection

Description
Potential Off by One Error in Loops\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=73
Status	New

The buffer allocated by <= in AndroidUSBCamera/lame.c at line 103 does not correctly account for the actual size of the value, resulting in an incorrect allocation that is off by one.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	118	118
Object	<=	<=

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_init_params_ppflt(lame_internal_flags * gfc)

```
....
118.          for (band = 0; band <= 31; band++) {
```

Potential Off by One Error in Loops\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=74
Status	New

The buffer allocated by <= in AndroidUSBCamera/lame.c at line 103 does not correctly account for the actual size of the value, resulting in an incorrect allocation that is off by one.

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	152	152
Object	<=	<=

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_init_params_ppflt(lame_internal_flags * gfc)

```
....
152.          for (band = 0; band <= 31; band++) {
```

Potential Off by One Error in Loops\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=75
Status	New

The buffer allocated by <= in AndroidUSBCamera/rdppm.c at line 307 does not correctly account for the actual size of the value, resulting in an incorrect allocation that is off by one.

	Source	Destination
File	AndroidUSBCamera/rdppm.c	AndroidUSBCamera/rdppm.c
Line	431	431
Object	<=	<=

Code Snippet
File Name AndroidUSBCamera/rdppm.c
Method start_input_ppm (j_compress_ptr cinfo, cjpeg_source_ptr sinfo)

```
....
431.      for (val = 0; val <= (long) maxval; val++) {
```

Arithmenic Operation On Boolean

Query Path:
CPP\Cx\CPP Low Visibility\Arithmenic Operation On Boolean Version:1

Categories

FISMA 2014: Audit And Accountability
NIST SP 800-53: SC-5 Denial of Service Protection (P1)

Description

Arithmenic Operation On Boolean\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=110
Status	New

	Source	Destination
File	AndroidUSBCamera/lame.c	AndroidUSBCamera/lame.c
Line	965	965
Object	BinaryExpr	BinaryExpr

Code Snippet
File Name AndroidUSBCamera/lame.c
Method lame_init_params(lame_global_flags * gfp)

```
....
965.      j = cfg->samplerate_index + (3 * cfg->version) + 6 * (cfg-
>samplerate_out < 16000);
```

Arithmenic Operation On Boolean\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=111
Status	New

	Source	Destination
File	AndroidUSBCamera/misctest.cpp	AndroidUSBCamera/misctest.cpp
Line	147	147
Object	BinaryExpr	BinaryExpr

Code Snippet
File Name AndroidUSBCamera/misctest.cpp

Method	inline unsigned CountDecimalDigit_fast(unsigned n) {

```
....
147.      return t - (n < powers_of_10[t]) + 1;
```

Arithmenic Operation On Boolean\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=112
Status	New

	Source	Destination
File	AndroidUSBCamera/misctest.cpp	AndroidUSBCamera/misctest.cpp
Line	192	192
Object	BinaryExpr	BinaryExpr

Code Snippet	
File Name	AndroidUSBCamera/misctest.cpp
Method	inline unsigned CountDecimalDigit64_fast(uint64_t n) {

```
....
192.      return t - (n < powers_of_10[t]) + 1;
```

Unchecked Array Index

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

Categories

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

Description

Unchecked Array Index\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=113
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	252	252
Object	rindex	rindex

Code Snippet	
File Name	AndroidUSBCamera/tjbench.c
Method	int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
252.                                    dstbuf[rindex]=abs(dstbuf[rindex]-
y);
```

Unchecked Array Index\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=114
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	253	253
Object	gindex	gindex

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
253.                                    dstbuf[gindex]=abs(dstbuf[gindex]-
y);
```

Unchecked Array Index\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=115
Status	New

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	254	254
Object	bindex	bindex

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int decomp(unsigned char *srcbuf, unsigned char **jpegbuf,

```
....
254.                                    dstbuf[bindex]=abs(dstbuf[bindex]-
y);
```

Heuristic Buffer Overflow malloc

Query Path:
CPP\Cx\CPP Heuristic\Heuristic Buffer Overflow malloc Version:0

Categories

PCI DSS v3.2: PCI DSS (3.2) - 6.5.2 - Buffer overflows
NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description
Heuristic Buffer Overflow malloc\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050091&projectid=50081&pathid=103
Status	New

The size of the buffer used by decompTest in srcsize, at line 480 of AndroidUSBCamera/tjbench.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that main passes to argv, at line 771 of AndroidUSBCamera/tjbench.c, to overwrite the target buffer.

	Source	Destination
File	AndroidUSBCamera/tjbench.c	AndroidUSBCamera/tjbench.c
Line	771	497
Object	argv	srcsize

Code Snippet
File Name AndroidUSBCamera/tjbench.c
Method int main(int argc, char *argv[])

```
....
771.  int main(int argc, char *argv[])
```

File Name AndroidUSBCamera/tjbench.c
Method int decompTest(char *filename)

```
....
497.        if((srcbuf=(unsigned char *)malloc(srcsize))==NULL)
```

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

CPP

Overflowing Buffers

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

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

Checked Buffers

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

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

Buffer Overflow OutOfBound

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

Source Code Examples

Buffer Overflow Indexes

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 boundedcpy

Risk

What might happen

Allowing tainted inputs to set the size of how many bytes to copy from source to destination may cause memory corruption, unexpected behavior, instability and data leakage. In some cases, such as when additional and specific areas of memory are also controlled by user input, it may result in code execution.

Cause

How does it happen

Should the size of the amount of bytes to copy from source to destination be greater than the size of the destination, an overflow will occur, and memory beyond the intended buffer will get overwritten. Since this size value is derived from user input, the user may provide an invalid and dangerous buffer size.

General Recommendations

How to avoid it

- Do not trust memory allocation sizes provided by the user; derive them from the copied values instead.
- If memory allocation by a provided value is absolutely required, restrict this size to safe values only. Specifically ensure that this value does not exceed the destination buffer's size.

Source Code Examples

CPP

Size Parameter is Influenced by User Input

```
char dest_buf[10];
memset(dest_buf, '\0', sizeof(dest_buf));
strncpy(dest_buf, src_buf, size); //Assuming size is provided by user input
```

Validating Destination Buffer Length

```
char dest_buf[10];
memset(dest_buf, '\0', sizeof(dest_buf));
if (size < sizeof(dest_buf) && sizeof(src_buf) >= size) //Assuming size is provided by user
input
{
      strncpy(dest_buf, src_buf, size);
}
else
{
      //...
}
```

Off by One Error in Arrays

Risk

What might happen

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

Cause

How does it happen

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

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

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

General Recommendations

How to avoid it

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

Source Code Examples

Buffer Overflow StrcpyStrcat

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 IndexFromInput

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

Divide By Zero

Risk

What might happen

When a program divides a number by zero, an exception will be raised. If this exception is not handled by the application, unexpected results may occur, including crashing the application. This can be considered a DoS (Denial of Service) attack, if an external user has control of the value of the denominator or can cause this error to occur.

Cause

How does it happen

The program receives an unexpected value, and uses it for division without filtering, validation, or verifying that the value is not zero. The application does not explicitly handle this error or prevent division by zero from occuring.

General Recommendations

How to avoid it

- Before dividing by an unknown value, validate the number and explicitly ensure it does not evaluate to zero.
- Validate all untrusted input from all sources, in particular verifying that it is not zero before dividing with it.
- Verify output of methods, calculations, dictionary lookups, and so on, and ensure it is not zero before dividing with the result.
- Ensure divide-by-zero errors are caught and handled appropriately.

Source Code Examples

Java

Divide by Zero

```
public float getAverage(HttpServletRequest req) {
      int total = Integer.parseInt(req.getParameter("total"));
      int count = Integer.parseInt(req.getParameter("count"));

      return total / count;
}
```

Checked Division

```
public float getAverage(HttpServletRequest req) {
      int total = Integer.parseInt(req.getParameter("total"));
      int count = Integer.parseInt(req.getParameter("count"));
```

```
	if (count > 0)
		return total / count;
	else
		return 0;
}
```

Buffer Overflow AddressOfLocalVarReturned

Risk

What might happen

A use after free error will cause code to use an area of memory previously assigned with a specific value, which has since been freed and may have been overwritten by another value. This error will likely cause unexpected behavior, memory corruption and crash errors. In some cases where the freed and used section of memory is used to determine execution flow, and the error can be induced by an attacker, this may result in execution of malicious code.

Cause

How does it happen

Pointers to variables allow code to have an address with a set size to a dynamically allocated variable. Eventually, the pointer's destination may become free - either explicitly in code, such as when programmatically freeing this variable, or implicitly, such as when a local variable is returned - once it is returned, the variable's scope is released. Once freed, this memory will be re-used by the application, overwritten with new data. At this point, dereferencing this pointer will potentially resolve newly written and unexpected data.

General Recommendations

How to avoid it

- Do not return local variables or pointers
- Review code to ensure no flow allows use of a pointer after it has been explicitly freed

---

Source Code Examples

CPP

Use of Variable after It was Freed

```
free(input);
printf("%s", input);
```

Use of Pointer to Local Variable That Was Freed On Return

```
int* func1()
{
	int i;
	i = 1;
	return &i;
}

void func2()
```

```
{
	int j;
	j = 5;
}

//..
	int * i = func1();
	printf("%d\r\n", *i); // Output could be 1 or Segmentation Fault
	func2();
	printf("%d\r\n", *i); // Output is 5, which is j's value, as func2() overwrote data in
the stack
//..
```

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

Wrong Size t Allocation

Risk

What might happen

Incorrect allocation of memory may result in unexpected behavior by either overwriting sections of memory with unexpected values. Under certain conditions where both an incorrect allocation of memory and the values being written can be controlled by an attacker, such an issue may result in execution of malicious code.

Cause

How does it happen

Some memory allocation functions require a size value to be provided as a parameter. The allocated size should be derived from the provided value, by providing the length value of the intended source, multiplied by the size of that length. Failure to perform the correct arithmetic to obtain the exact size of the value will likely result in the source overflowing its destination.

General Recommendations

How to avoid it

- Always perform the correct arithmetic to determine size.
- Specifically for memory allocation, calculate the allocation size from the allocation source:
  - Derive the size value from the length of intended source to determine the amount of units to be processed.
  - Always programmatically consider the size of the each unit and their conversion to memory units - for example, by using sizeof() on the unit's type.
  - Memory allocation should be a multiplication of the amount of units being written, times the size of each unit.

Source Code Examples

CPP

Allocating and Assigning Memory without Sizeof Arithmetic

```
int *ptr;
ptr = (int*)malloc(5);
for (int i = 0; i < 5; i++)
{
      ptr[i] = i * 2 + 1;
}
```

Allocating and Assigning Memory with Sizeof Arithmetic

```
int *ptr;
ptr = (int*)malloc(5 * sizeof(int));
```

```
for (int i = 0; i < 5; i++)
{
      ptr[i] = i * 2 + 1;
}
```

Incorrect Arithmetic of Multi-Byte String Allocation

```
wchar_t * dest;
dest = (wchar_t *)malloc(wcslen(source) + 1); // Would not crash for a short "source"
wcscpy((wchar_t *)dest, source);
wprintf(L"Dest: %s\r\n", dest);
```

Correct Arithmetic of Multi-Byte String Allocation

```
wchar_t * dest;
dest = (wchar_t *)malloc((wcslen(source) + 1) * sizeof(wchar_t));
wcscpy((wchar_t *)dest, source);
wprintf(L"Dest: %s\r\n", dest);
```

Integer Overflow

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

---

Source Code Examples

Short Overflow

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

---

Source Code Examples

CPP

Unsafe Downsize Casting

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

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

    return total;
}
```

Safer Use of Proper Data Types

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

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

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

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

    return total;
}
```

Dangerous Functions

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

Source Code Examples

CPP

Buffer Overflow in gets()

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

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

Safe reading from user

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

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

Unsafe function for string copy

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

	return 0;
}
```

Safe string copy

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

	return 0;
}
```

Unsafe format string

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

Safe format string

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

Double Free

Weakness ID: 415 *(Weakness Variant)* **Status:** Draft

Description

Description Summary

The product calls free() twice on the same memory address, potentially leading to modification of unexpected memory locations.

Extended Description

When a program calls free() twice with the same argument, the program's memory management data structures become corrupted. This corruption can cause the program to crash or, in some circumstances, cause two later calls to malloc() to return the same pointer. If malloc() returns the same value twice and the program later gives the attacker control over the data that is written into this doubly-allocated memory, the program becomes vulnerable to a buffer overflow attack.

Alternate Terms

Double-free

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

C

C++

Common Consequences

Scope	Effect
Access Control	Doubly freeing memory may result in a write-what-where condition, allowing an attacker to execute arbitrary code.

Likelihood of Exploit

Low to Medium

Demonstrative Examples

Example 1

The following code shows a simple example of a double free vulnerability.

(Bad Code)

Example Language: **C**

```
char* ptr = (char*)malloc (SIZE);
...
if (abrt) {
free(ptr);
}
...
free(ptr);
```

Double free vulnerabilities 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

Although some double free vulnerabilities are not much more complicated than the previous example, most are spread out across hundreds of lines of code or even different files. Programmers seem particularly susceptible to freeing global variables

more than once.

Example 2

While contrived, this code should be exploitable on Linux distributions which do not ship with heap-chunk check summing turned on.

(Bad Code)

Example Language: **C**

```
#include <stdio.h>
#include <unistd.h>
#define BUFSIZE1 512
#define BUFSIZE2 ((BUFSIZE1/2) - 8)

int main(int argc, char **argv) {
char *buf1R1;
char *buf2R1;
char *buf1R2;
buf1R1 = (char *) malloc(BUFSIZE2);
buf2R1 = (char *) malloc(BUFSIZE2);
free(buf1R1);
free(buf2R1);
buf1R2 = (char *) malloc(BUFSIZE1);
strncpy(buf1R2, argv[1], BUFSIZE1-1);
free(buf2R1);
free(buf1R2);
}
```

Observed Examples

Reference	Description
CVE-2004-0642	Double free resultant from certain error conditions.
CVE-2004-0772	Double free resultant from certain error conditions.
CVE-2005-1689	Double free resultant from certain error conditions.
CVE-2003-0545	Double free from invalid ASN.1 encoding.
CVE-2003-1048	Double free from malformed GIF.
CVE-2005-0891	Double free from malformed GIF.
CVE-2002-0059	Double free from malformed compressed data.

Potential Mitigations

Phase: Architecture and Design

Choose a language that provides automatic memory management.

Phase: Implementation

Ensure that each allocation is freed only once. After freeing a chunk, set the pointer to NULL to ensure the pointer cannot be freed again. In complicated error conditions, be sure that clean-up routines respect the state of allocation properly. If the language is object oriented, ensure that object destructors delete each chunk of memory only once.

Phase: Implementation

Use a static analysis tool to find double free instances.

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	666	Operation on Resource in Wrong Phase of	**Research Concepts (primary)1000**

			Lifetime	
ChildOf	Weakness Class	675	Duplicate Operations on Resource	Research Concepts1000
ChildOf	Category	742	CERT C Secure Coding Section 08 - Memory Management (MEM)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
PeerOf	Weakness Base	123	Write-what-where Condition	Research Concepts1000
PeerOf	Weakness Base	416	Use After Free	Development Concepts699 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

Relationship Notes

This is usually resultant from another weakness, such as an unhandled error or race condition between threads. It could also be primary to weaknesses such as buffer overflows.

Affected Resources

- Memory

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			DFREE - Double-Free Vulnerability
7 Pernicious Kingdoms			Double Free
CLASP			Doubly freeing 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	MEM31-C		Free dynamically allocated memory exactly once

White Box Definitions

A weakness where code path has:

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

Maintenance Notes

It could be argued that Double Free would be most appropriately located as a child of "Use after Free", but "Use" and "Release" are considered to be distinct operations within vulnerability theory, therefore this is more accurately "Release of a Resource after Expiration or Release", which doesn't exist yet.

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 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, Description, Maintenance Notes, Relationships, Other Notes, Relationship Notes, Taxonomy Mappings		
2008-11-24	CWE Content Team	MITRE	Internal

	updated Relationships, Taxonomy Mappings		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Other Notes		

BACK TO TOP

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

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

Description

Description Summary

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

Extended Description

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

Terminology Notes

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

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

C

C++

Modes of Introduction

Memory leaks have two common and sometimes overlapping causes:

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

Common Consequences

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

Likelihood of Exploit

Medium

Demonstrative Examples

Example 1

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

(Bad Code)

Example Language: **C**

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

return NULL;
}
```

```
return buf;
}
```

Example 2

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

(Bad Code)

Example Language: **C**

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

free(foo);
}
int main() {

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

Observed Examples

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

Potential Mitigations

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

Phase: Architecture and Design

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

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

Relationships

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

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

Relationship Notes

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

Affected Resources

- Memory

Functional Areas

- Memory management

Taxonomy Mappings

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

White Box Definitions

A weakness where the code path has:

1. start statement that allocates dynamically allocated memory resource

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

Where "loses" is defined through the following scenarios:

1. identity of the dynamic allocated memory resource never obtained

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

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

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

References

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

Content History

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

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

Previous Entry Names

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

BACK TO TOP

Potential Path Traversal

Risk

What might happen

An attacker could define any arbitrary file path for the application to use, potentially leading to:

- Stealing sensitive files, such as configuration or system files
- Overwriting files such as program binaries, configuration files, or system files
- Deleting critical files, causing a denial of service (DoS).

Cause

How does it happen

The application uses user input in the file path for accessing files on the application server's local disk. This enables an attacker to arbitrarily determine the file path.

General Recommendations

How to avoid it

1. Ideally, avoid depending on user input for file selection.
2. Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns. Check for:
   - Data type
   - Size
   - Range
   - Format
   - Expected values
3. Accept user input only for the filename, not for the path and folders.
4. Ensure that file path is fully canonicalized.
5. Explicitly limit the application to using a designated folder that separate from the applications binary folder.
6. Restrict the privileges of the application's OS user to necessary files and folders. The application should not be able to write to the application binary folder, and should not read anything outside of the application folder and data folder.

Source Code Examples

CSharp

Using unvalidated user input as the file name may enable the user to access arbitrary files on the server local disk

```
public class PathTraversal
{
        private void foo(TextBox textbox1)
    {
                string fileNum = textbox1.Text;
                string path = "c:\files\file" + fileNum;
                FileStream f = new FileStream(path, FileMode.Open);
                byte[] output = new byte[10];
                f.Read(output,0, 10);
```

```
        }
}
```

Potentially hazardous characters are removed from the user input before use

```
public class PathTraversalFixed
{
        private void foo(TextBox textbox1)
    {
                string fileNum = textbox1.Text.Replace("\", "").Replace("..", "");
         string path = "c:\files\file" + fileNum;
                FileStream f = new FileStream(path, FileMode.Open);
                byte[] output = new byte[10];
                f.Read(output,0, 10);
        }
}
```

Java

Using unvalidated user input as the file name may enable the user to access arbitrary files on the server local disk

```
public class Absolute_Path_Traversal {
    public static void main(String[] args) {
        Scanner userInputScanner = new Scanner(System.in);
        System.out.print("\nEnter file name: ");
        String name = userInputScanner.nextLine();
        String path = "c:\files\file" + name;
        try {
            BufferedReader reader = new BufferedReader(new FileReader(path));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Potentially hazardous characters are removed from the user input before use

```
public class Absolute_Path_Traversal_Fixed {
    public static void main(String[] args) {
        Scanner userInputScanner = new Scanner(System.in);
        System.out.print("\nEnter file name: ");
        String name = userInputScanner.nextLine();
        name = name.replace("/", "").replace("..", "");
        String path = "c:\files\file" + name;
        try {
            BufferedReader reader = new BufferedReader(new FileReader(path));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Unchecked Return Value

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

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

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

Source Code Examples

CPP

Unchecked Memory Allocation

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

Safer Memory Allocation

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

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

Use of sizeof() on a Pointer Type

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

Description

Description Summary

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

Time of Introduction

- Implementation

Applicable Platforms

Languages

C

C++

Common Consequences

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

Likelihood of Exploit

High

Demonstrative Examples

Example 1

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

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

(Bad Code)

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

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

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

(Good Code)

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

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

Example 2

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

(Bad Code)

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

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

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

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

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

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

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

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

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

(Attack)

```
pass5
passABCDEFGH
passWORD
```

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

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

Potential Mitigations

Phase: Implementation

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

Other Notes

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

Weakness Ordinalities

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

Relationships

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

Taxonomy Mappings

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

White Box Definitions

A weakness where code path has:

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

References

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

Content History

Submissions

Submission Date	Submitter	Organization	Source
	CLASP		Externally Mined

Modifications

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

BACK TO TOP

Potential Off by One Error in Loops

Risk

What might happen

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

Cause

How does it happen

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

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

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

General Recommendations

How to avoid it

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

Source Code Examples

CPP

Off-By-One in For Loop

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

```
}
```

Proper Iteration in For Loop

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

Off-By-One in strncat

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

Heuristic Buffer Overflow malloc

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

Indicator of Poor Code Quality

Weakness ID: 398 *(Weakness Class)* **Status:** Draft

Description

Description Summary

The code has features that do not directly introduce a weakness or vulnerability, but indicate that the product has not been carefully developed or maintained.

Extended Description

Programs are more likely to be secure when good development practices are followed. If a program is complex, difficult to maintain, not portable, or shows evidence of neglect, then there is a higher likelihood that weaknesses are buried in the code.

Time of Introduction

- Architecture and Design
- Implementation

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	18	Source Code	**Development Concepts (primary)699**
ChildOf	Weakness Class	710	Coding Standards Violation	**Research Concepts (primary)1000**
ParentOf	Weakness Variant	107	Struts: Unused Validation Form	**Research Concepts (primary)1000**
ParentOf	Weakness Variant	110	Struts: Validator Without Form Field	**Research Concepts (primary)1000**
ParentOf	Category	399	Resource Management Errors	**Development Concepts (primary)699**
ParentOf	Weakness Base	401	Failure to Release Memory Before Removing Last Reference ('Memory Leak')	**Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Base	404	Improper Resource Shutdown or Release	Development Concepts699 **Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Variant	415	Double Free	**Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Base	416	Use After Free	**Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Variant	457	Use of Uninitialized Variable	**Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Base	474	Use of Function with Inconsistent Implementations	**Development Concepts (primary)699 Seven Pernicious Kingdoms (primary)700 Research Concepts (primary)1000**
ParentOf	Weakness Base	475	Undefined Behavior for Input to API	**Development Concepts (primary)699 Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Base	476	NULL Pointer	**Development**

			Dereference	**Concepts (primary)699 Seven Pernicious Kingdoms (primary)700 Research Concepts (primary)1000**
ParentOf	Weakness Base	477	Use of Obsolete Functions	**Development Concepts (primary)699 Seven Pernicious Kingdoms (primary)700 Research Concepts (primary)1000**
ParentOf	Weakness Variant	478	Missing Default Case in Switch Statement	**Development Concepts (primary)699**
ParentOf	Weakness Variant	479	Unsafe Function Call from a Signal Handler	**Development Concepts (primary)699**
ParentOf	Weakness Variant	483	Incorrect Block Delimitation	**Development Concepts (primary)699**
ParentOf	Weakness Base	484	Omitted Break Statement in Switch	**Development Concepts (primary)699** Research Concepts1000
ParentOf	Weakness Variant	546	Suspicious Comment	**Development Concepts (primary)699 Research Concepts (primary)1000**
ParentOf	Weakness Variant	547	Use of Hard-coded, Security-relevant Constants	**Development Concepts (primary)699 Research Concepts (primary)1000**
ParentOf	Weakness Variant	561	Dead Code	**Development Concepts (primary)699 Research Concepts (primary)1000**
ParentOf	Weakness Base	562	Return of Stack Variable Address	**Development Concepts (primary)699** Research Concepts1000
ParentOf	Weakness Variant	563	Unused Variable	**Development Concepts (primary)699 Research Concepts (primary)1000**
ParentOf	Category	569	Expression Issues	**Development Concepts (primary)699**
ParentOf	Weakness Variant	585	Empty Synchronized Block	**Development Concepts (primary)699 Research Concepts (primary)1000**
ParentOf	Weakness Variant	586	Explicit Call to Finalize()	**Development Concepts (primary)699**
ParentOf	Weakness Variant	617	Reachable Assertion	**Development Concepts (primary)699**
ParentOf	Weakness Base	676	Use of Potentially Dangerous Function	**Development Concepts (primary)699 Research Concepts (primary)1000**
MemberOf	View	700	Seven Pernicious Kingdoms	**Seven Pernicious Kingdoms (primary)700**

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name

7 Pernicious Kingdoms			Code Quality

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		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Description, Relationships, Taxonomy Mappings		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Relationships		
Previous Entry Names			
Change Date	**Previous Entry Name**		
2008-04-11	Code Quality		

BACK TO TOP

Improper Validation of Array Index

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

Description

Description Summary

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

Alternate Terms

out-of-bounds array index

index-out-of-range

array index underflow

Time of Introduction

- Implementation

Applicable Platforms

Languages

C: *(Often)*

C++: *(Often)*

Language-independent

Common Consequences

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

Likelihood of Exploit

High

Detection Methods

Automated Static Analysis

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

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

Effectiveness: High

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

Automated Dynamic Analysis

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

Black Box

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

Demonstrative Examples

Example 1

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

(Bad Code)

Example Language: **C**

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

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

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

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

(Good Code)

Example Language: **C**

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

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

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

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

Example 2

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

(Bad Code)

Example Language: **Java**

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

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

Example 3

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

(Bad Code)

Example Language: **Java**

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

String productSummary = new String("");

try {
String productSummary = getProductSummary(index);

} catch (Exception ex) {...}

return productSummary;
}

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

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

(Good Code)

Example Language: **Java**

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

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

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

} catch (Exception ex) {...}

return productSummary;
}

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

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

return productSummary;
}
```

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

(Good Code)

Example Language: **Java**

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

Observed Examples

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

Potential Mitigations

Phase: Architecture and Design

Strategies: Input Validation; Libraries or Frameworks

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

Phase: Architecture and Design

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

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

Phase: Requirements

Strategy: Language Selection

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

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

Phase: Implementation

Strategy: Input Validation

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

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

Phase: Implementation

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

Weakness Ordinalities

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

Relationships

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

Theoretical Notes

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

Affected Resources

- Memory

f Causal Nature

Explicit

Taxonomy Mappings

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

Related Attack Patterns

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

References

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

Content History

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

BACK TO TOP

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

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

Exposure of System Data to Unauthorized Control Sphere

Risk

What might happen

System data can provide attackers with valuable insights on systems and services they are targeting - any type of system data, from service version to operating system fingerprints, can assist attackers to hone their attack, correlate data with known vulnerabilities or focus efforts on developing new attacks against specific technologies.

Cause

How does it happen

System data is read and subsequently exposed where it might be read by untrusted entities.

General Recommendations

How to avoid it

Consider the implications of exposure of the specified input, and expected level of access to the specified output. If not required, consider removing this code, or modifying exposed information to exclude potentially sensitive system data.

Source Code Examples

Java

Leaking Environment Variables in JSP Web-Page

```
String envVarValue = System.getenv(envVar);
if (envVarValue == null) {
    out.println("Environment variable is not defined:");
    out.println(System.getenv());
} else {
    //[..]
};
```

TOCTOU

Risk

What might happen

At best, a Race Condition may cause errors in accuracy, overidden values or unexpected behavior that may result in denial-of-service. At worst, it may allow attackers to retrieve data or bypass security processes by replaying a controllable Race Condition until it plays out in their favor.

Cause

How does it happen

Race Conditions occur when a public, single instance of a resource is used by multiple concurrent logical processes. If the these logical processes attempt to retrieve and update the resource without a timely management system, such as a lock, a Race Condition will occur.

An example for when a Race Condition occurs is a resource that may return a certain value to a process for further editing, and then updated by a second process, resulting in the original process' data no longer being valid. Once the original process edits and updates the incorrect value back into the resource, the second process' update has been overwritten and lost.

General Recommendations

How to avoid it

When sharing resources between concurrent processes across the application ensure that these resources are either thread-safe, or implement a locking mechanism to ensure expected concurrent activity.

Source Code Examples

Java

Different Threads Increment and Decrement The Same Counter Repeatedly, Resulting in a Race Condition

```
	public static int counter = 0;
	public static void start() throws InterruptedException {
		incrementCounter ic;
		decrementCounter dc;
		while(counter == 0) {
			counter = 0;
			ic = new incrementCounter();
			dc = new decrementCounter();
			ic.start();
			dc.start();
			ic.join();
			dc.join();
		}
		System.out.println(counter); //Will stop and return either -1 or 1 due to race
condition over counter
	}

	public static class incrementCounter extends Thread {
	    public void run() {
		counter++;
	    }
```

```
    }

    public static class decrementCounter extends Thread {
        public void run() {
           counter--;
        }
    }
```

Different Threads Increment and Decrement The Same Thread-Safe Counter Repeatedly, Never Resulting in a Race Condition

```
    public static int counter = 0;
    public static Object lock = new Object();

    public static void start() throws InterruptedException {
           incrementCounter ic;
           decrementCounter dc;
           while(counter == 0) { // because of proper locking, this condition is never false
                  counter = 0;
                  ic = new incrementCounter();
                  dc = new decrementCounter();
                  ic.start();
                  dc.start();
                  ic.join();
                  dc.join();
           }
           System.out.println(counter); // Never reached
    }

    public static class incrementCounter extends Thread {
        public void run() {
           synchronized (lock) {
                  counter++;
           }
        }
    }

    public static class decrementCounter extends Thread {
        public void run() {
           synchronized (lock) {
                  counter--;
           }
        }
    }
```

Scanned Languages

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