vul_files_57 Scan Report

Project Name	vul_files_57
Scan Start	Wednesday, January 8, 2025 8:14:38 PM
Preset	Checkmarx Default
Scan Time	02h:03m:24s
Lines Of Code Scanned	299312
Files Scanned	460
Report Creation Time	Wednesday, January 8, 2025 11:49:46 PM
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059
Team	CxServer
Checkmarx Version	8.7.0
Scan Type	Full
Source Origin	LocalPath
Density	1/10000 (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

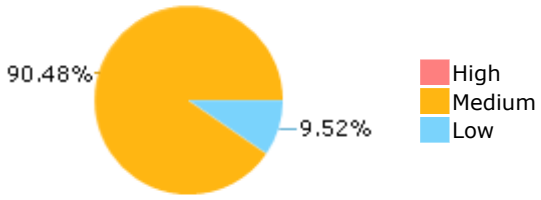
Results Limit

Results limit per query was set to 50

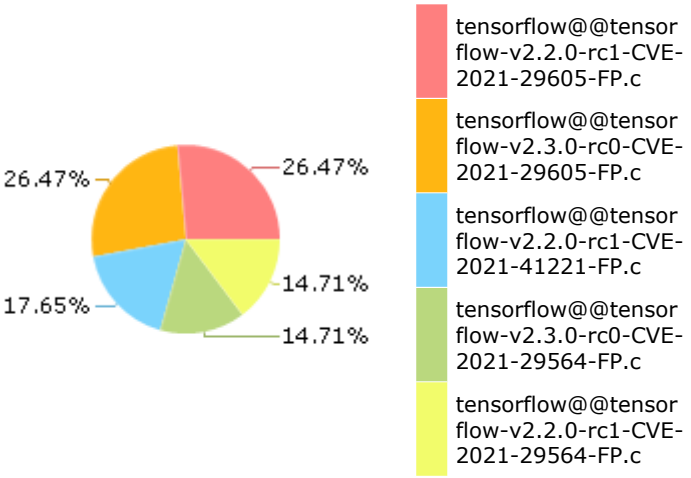
Selected Queries

Selected queries are listed in Result Summary

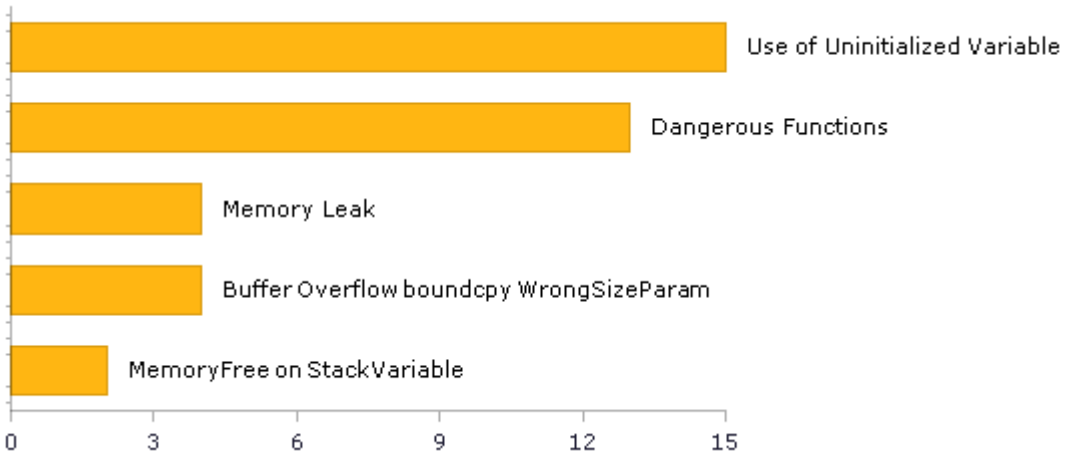
Result Summary



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	4	4
A2-Broken Authentication	App. Specific	EASY	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A3-Sensitive Data Exposure	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App. Specific	0	0
A4-XML External Entities (XXE)	App. Specific	AVERAGE	COMMON	EASY	SEVERE	App. Specific	0	0
A5-Broken Access Control*	App. Specific	AVERAGE	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A6-Security Misconfiguration	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A7-Cross-Site Scripting (XSS)	App. Specific	EASY	WIDESPREAD	EASY	MODERATE	App. Specific	0	0
A8-Insecure Deserialization	App. Specific	DIFFICULT	COMMON	AVERAGE	SEVERE	App. Specific	0	0
A9-Using Components with Known Vulnerabilities*	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	MODERATE	App. Specific	13	13
A10-Insufficient Logging & Monitoring	App. Specific	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	App. Specific	0	0

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

Scan Summary - OWASP Top 10 2013

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

Category	Threat Agent	Attack Vectors	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	AVERAGE	SEVERE	ALL DATA	0	0
A2-Broken Authentication and Session Management	EXTERNAL, INTERNAL USERS	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	AFFECTED DATA AND FUNCTIONS	0	0
A3-Cross-Site Scripting (XSS)	EXTERNAL, INTERNAL, ADMIN USERS	AVERAGE	VERY WIDESPREAD	EASY	MODERATE	AFFECTED DATA AND SYSTEM	0	0
A4-Insecure Direct Object References	SYSTEM USERS	EASY	COMMON	EASY	MODERATE	EXPOSED DATA	0	0
A5-Security Misconfiguration	EXTERNAL, INTERNAL, ADMIN USERS	EASY	COMMON	EASY	MODERATE	ALL DATA AND SYSTEM	0	0
A6-Sensitive Data Exposure	EXTERNAL, INTERNAL, ADMIN USERS, USERS BROWSERS	DIFFICULT	UNCOMMON	AVERAGE	SEVERE	EXPOSED DATA	0	0
A7-Missing Function Level Access Control*	EXTERNAL, INTERNAL USERS	EASY	COMMON	AVERAGE	MODERATE	EXPOSED DATA AND FUNCTIONS	0	0
A8-Cross-Site Request Forgery (CSRF)	USERS BROWSERS	AVERAGE	COMMON	EASY	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0
A9-Using Components with Known Vulnerabilities*	EXTERNAL USERS, AUTOMATED TOOLS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	13	13
A10-Unvalidated Redirects and Forwards	USERS BROWSERS	AVERAGE	WIDESPREAD	DIFFICULT	MODERATE	AFFECTED DATA AND FUNCTIONS	0	0

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

Scan Summary - PCI DSS v3.2

Category	Issues Found	Best Fix Locations
PCI DSS (3.2) - 6.5.1 - Injection flaws - particularly SQL injection	0	0
PCI DSS (3.2) - 6.5.2 - Buffer overflows	4	4
PCI DSS (3.2) - 6.5.3 - Insecure cryptographic storage	0	0
PCI DSS (3.2) - 6.5.4 - Insecure communications	0	0
PCI DSS (3.2) - 6.5.5 - Improper error handling*	0	0
PCI DSS (3.2) - 6.5.7 - Cross-site scripting (XSS)	0	0
PCI DSS (3.2) - 6.5.8 - Improper access control	0	0
PCI DSS (3.2) - 6.5.9 - Cross-site request forgery	0	0
PCI DSS (3.2) - 6.5.10 - Broken authentication and session management	0	0

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

Scan Summary - FISMA 2014

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

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

Scan Summary - NIST SP 800-53

Category	Issues Found	Best Fix Locations
AC-12 Session Termination (P2)	0	0
AC-3 Access Enforcement (P1)	0	0
AC-4 Information Flow Enforcement (P1)	0	0
AC-6 Least Privilege (P1)	0	0
AU-9 Protection of Audit Information (P1)	0	0
CM-6 Configuration Settings (P2)	0	0
IA-5 Authenticator Management (P1)	0	0
IA-6 Authenticator Feedback (P2)	0	0
IA-8 Identification and Authentication (Non-Organizational Users) (P1)	0	0
SC-12 Cryptographic Key Establishment and Management (P1)	0	0
SC-13 Cryptographic Protection (P1)	0	0
SC-17 Public Key Infrastructure Certificates (P1)	0	0
SC-18 Mobile Code (P2)	0	0
SC-23 Session Authenticity (P1)*	0	0
SC-28 Protection of Information at Rest (P1)	0	0
SC-4 Information in Shared Resources (P1)	0	0
SC-5 Denial of Service Protection (P1)*	19	7
SC-8 Transmission Confidentiality and Integrity (P1)	0	0
SI-10 Information Input Validation (P1)*	0	0
SI-11 Error Handling (P2)*	4	4
SI-15 Information Output Filtering (P0)	0	0
SI-16 Memory Protection (P1)	0	0

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

Scan Summary - OWASP Mobile Top 10 2016

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

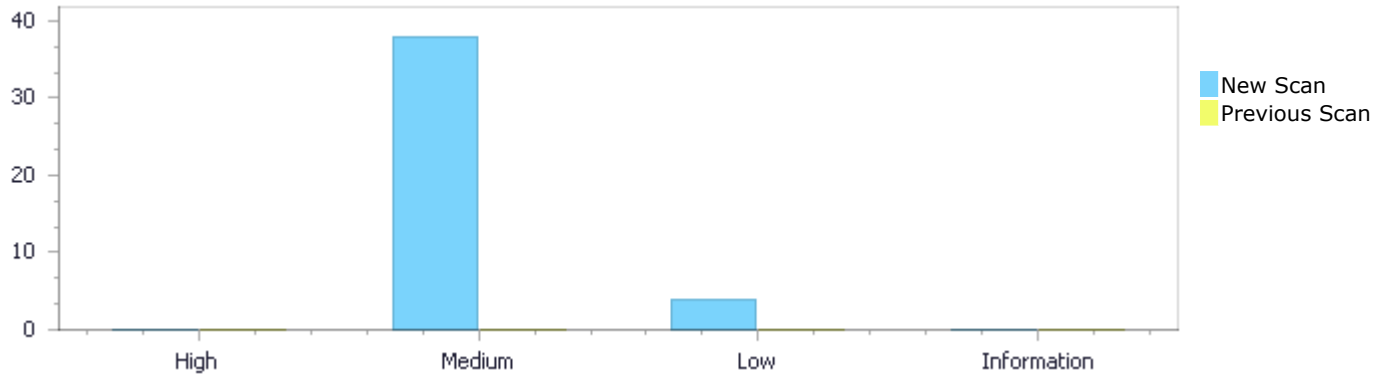
	modify the application's data and resources. This can provide the attacker a direct method of subverting the intended use of the software for personal or monetary gain.		
M9-Reverse Engineering	This category includes analysis of the final core binary to determine its source code, libraries, algorithms,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

Scan Summary - Custom

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

Results Distribution By Status First scan of the project

	High	Medium	Low	Information	Total
New Issues	0	38	4	0	42
Recurrent Issues	0	0	0	0	0
Total	0	38	4	0	42
Fixed Issues	0	0	0	0	0



Results Distribution By State

	High	Medium	Low	Information	Total
Confirmed	0	0	0	0	0
Not Exploitable	0	0	0	0	0
To Verify	0	38	4	0	42
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	0	38	4	0	42

Result Summary

Vulnerability Type	Occurrences	Severity
Use of Uninitialized Variable	15	Medium
Dangerous Functions	13	Medium
Buffer Overflow boundcpy WrongSizeParam	4	Medium
Memory Leak	4	Medium
MemoryFree on StackVariable	2	Medium

Unchecked Return Value	4	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c	7
tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c	7
tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c	6
tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c	5
tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c	5
tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c	5
tensorflow@@tensorflow-v2.16.0-rc0-CVE-2020-5215-FP.c	1
tensorflow@@tensorflow-v2.17.0-rc0-CVE-2020-5215-FP.c	1
tensorflow@@tensorflow-v2.3.0-rc0-CVE-2020-5215-FP.c	1

Scan Results Details

Use of Uninitialized Variable

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

Categories

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

Description

Use of Uninitialized Variable\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=28
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Line	233	190
Object	normalize_	normalize_

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Method void Compute(OpKernelContext* ctx) override {

```
....
190.          if (normalize_) output_t(loc) /= truth_seq.size();
```

Use of Uninitialized Variable\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=29
Status	New

	Source	Destination

File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Line	233	198
Object	normalize_	normalize_

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Method void Compute(OpKernelContext* ctx) override {

```
....
198.          if (normalize_ && output_t(loc) != 0.0f) {
```

Use of Uninitialized Variable\Path 3:

Severity Medium
Result State To Verify
Online Results http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=30
Status New

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Line	233	205
Object	normalize_	normalize_

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Method void Compute(OpKernelContext* ctx) override {

```
....
205.          output_t(loc) = (normalize_) ? 1.0 : truth_seq.size();
```

Use of Uninitialized Variable\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=31
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Line	233	216
Object	normalize_	normalize_

Code Snippet

File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c

Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c

Method void Compute(OpKernelContext* ctx) override {

```
....
216.        if (normalize_ && output_t(loc) != 0.0f) {
```

Use of Uninitialized Variable\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=32
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c
Line	233	227
Object	normalize_	normalize_

Code Snippet

File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c

Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29564-FP.c

Method void Compute(OpKernelContext* ctx) override {

```
....
227.        output_t(loc) = (normalize_) ? 1.0 : truth_seq.size();
```

Use of Uninitialized Variable\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=33
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c
Line	233	190
Object	normalize_	normalize_

Code Snippet

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c

Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c

Method void Compute(OpKernelContext* ctx) override {

```
....
190.          if (normalize_) output_t(loc) /= truth_seq.size();
```

Use of Uninitialized Variable\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=34
Status	New

Source	Destination

File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c
Line	233	198
Object	normalize_	normalize_

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c
Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c
Method void Compute(OpKernelContext* ctx) override {

```
....
198.          if (normalize_ && output_t(loc) != 0.0f) {
```

Use of Uninitialized Variable\Path 8:

Severity Medium
Result State To Verify
Online Results http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=35
Status New

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c
Line	233	205
Object	normalize_	normalize_

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c
Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c
Method void Compute(OpKernelContext* ctx) override {

```
....
205.          output_t(loc) = (normalize_) ? 1.0 : truth_seq.size();
```

Use of Uninitialized Variable\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=36
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c
Line	233	216
Object	normalize_	normalize_

Code Snippet

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c

Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c

Method void Compute(OpKernelContext* ctx) override {

```
....
216.        if (normalize_ && output_t(loc) != 0.0f) {
```

Use of Uninitialized Variable\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=37
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c
Line	233	227
Object	normalize_	normalize_

Code Snippet

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c

Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29564-FP.c

Method void Compute(OpKernelContext* ctx) override {

```
....
227.        output_t(loc) = (normalize_) ? 1.0 : truth_seq.size();
```

Use of Uninitialized Variable\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=38
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c
Line	233	190
Object	normalize_	normalize_

Code Snippet

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c

Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c

Method void Compute(OpKernelContext* ctx) override {

```
....
190.          if (normalize_) output_t(loc) /= truth_seq.size();
```

Use of Uninitialized Variable\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=39
Status	New

Source	Destination

File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c
Line	233	198
Object	normalize_	normalize_

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c
Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c
Method void Compute(OpKernelContext* ctx) override {

```
....
198.          if (normalize_ && output_t(loc) != 0.0f) {
```

Use of Uninitialized Variable\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=40
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c
Line	233	205
Object	normalize_	normalize_

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c
Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c
Method void Compute(OpKernelContext* ctx) override {

```
....
205.          output_t(loc) = (normalize_) ? 1.0 : truth_seq.size();
```

Use of Uninitialized Variable\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=41
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c
Line	233	216
Object	normalize_	normalize_

Code Snippet

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c

Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c

Method void Compute(OpKernelContext* ctx) override {

```
....
216.        if (normalize_ && output_t(loc) != 0.0f) {
```

Use of Uninitialized Variable\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=42
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c
Line	233	227
Object	normalize_	normalize_

Code Snippet

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c

Method bool normalize_;

```
....
233.    bool normalize_;
```

File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2022-29208-FP.c

Method void Compute(OpKernelContext* ctx) override {

```
....
227.        output_t(loc) = (normalize_) ? 1.0 : truth_seq.size();
```

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-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=11
Status	New

The dangerous function, memcpy, was found in use at line 52 in tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Line	56	56
Object	memcpy	memcpy

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Method TfLiteIntArray* TfLiteIntArrayCopy(const TfLiteIntArray* src) {

```
....
56.      memcpy(ret->data, src->data, src->size * sizeof(int));
```

Dangerous Functions\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30

	059&pathid=12
Status	New

The dangerous function, memcpy, was found in use at line 52 in tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Line	56	56
Object	memcpy	memcpy

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Method TfLiteIntArray* TfLiteIntArrayCopy(const TfLiteIntArray* src) {

```
....
56.      memcpy(ret->data, src->data, src->size * sizeof(int));
```

Dangerous Functions\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=13
Status	New

The dangerous function, StrCat, was found in use at line 713 in tensorflow@@tensorflow-v2.16.0-rc0-CVE-2020-5215-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.16.0-rc0-CVE-2020-5215-FP.c	tensorflow@@tensorflow-v2.16.0-rc0-CVE-2020-5215-FP.c
Line	754	754
Object	StrCat	StrCat

Code Snippet
File Name tensorflow@@tensorflow-v2.16.0-rc0-CVE-2020-5215-FP.c
Method TFE_TensorHandle* PySeqToTFE_TensorHandle(TFE_Context* ctx, PyObject* obj,

```
....
754.              tensorflow::strings::StrCat("Invalid dtype argument
value ", dtype)
```

Dangerous Functions\Path 4:

Severity	Medium
Result State	To Verify

Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=14
Status	New

The dangerous function, StrCat, was found in use at line 713 in tensorflow@@tensorflow-v2.17.0-rc0-CVE-2020-5215-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.17.0-rc0-CVE-2020-5215-FP.c	tensorflow@@tensorflow-v2.17.0-rc0-CVE-2020-5215-FP.c
Line	754	754
Object	StrCat	StrCat

Code Snippet
File Name tensorflow@@tensorflow-v2.17.0-rc0-CVE-2020-5215-FP.c
Method TFE_TensorHandle* PySeqToTFE_TensorHandle(TFE_Context* ctx, PyObject* obj,

```
....
754.              tensorflow::strings::StrCat("Invalid dtype argument
value ", dtype)
```

Dangerous Functions\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=15
Status	New

The dangerous function, StrCat, was found in use at line 105 in tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Line	120	120
Object	StrCat	StrCat

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Method TEST(CudnnRNNOpsTest, ForwardV3Lstm_ShapeFn) {

```
....
120.      return strings::StrCat("[", absl::StrJoin(v, ","), "]");
```

Dangerous Functions\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=16
Status	New

The dangerous function, StrCat, was found in use at line 105 in tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Line	122	122
Object	StrCat	StrCat

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Method TEST(CudnnRNNOpsTest, ForwardV3Lstm_ShapeFn) {

```
....
122.    string input_shapes_desc = strings::StrCat(
```

Dangerous Functions\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=17
Status	New

The dangerous function, StrCat, was found in use at line 41 in tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Line	53	53
Object	StrCat	StrCat

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Method TEST(CudnnRNNOpsTest, ForwardLstm_ShapeFn) {

```
....
53.      return strings::StrCat("[", absl::StrJoin(v, ","), "]");
```

Dangerous Functions\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=18
Status	New

The dangerous function, StrCat, was found in use at line 41 in tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Line	55	55
Object	StrCat	StrCat

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Method TEST(CudnnRNNOpsTest, ForwardLstm_ShapeFn) {

```
....
55.    string input_shapes_desc = strings::StrCat(
```

Dangerous Functions\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=19
Status	New

The dangerous function, StrCat, was found in use at line 73 in tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Line	85	85
Object	StrCat	StrCat

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Method TEST(CudnnRNNOpsTest, ForwardV2Lstm_ShapeFn) {

```
....
85.      return strings::StrCat("[", absl::StrJoin(v, ","), "]");
```

Dangerous Functions\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=20
Status	New

The dangerous function, StrCat, was found in use at line 73 in tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Line	87	87
Object	StrCat	StrCat

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-41221-FP.c
Method TEST(CudnnRNNOpsTest, ForwardV2Lstm_ShapeFn) {

```
....
87.    string input_shapes_desc = strings::StrCat(
```

Dangerous Functions\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=21
Status	New

The dangerous function, StrCat, was found in use at line 710 in tensorflow@@tensorflow-v2.3.0-rc0-CVE-2020-5215-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2020-5215-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2020-5215-FP.c
Line	740	740
Object	StrCat	StrCat

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2020-5215-FP.c
Method TFE_TensorHandle* PySeqToTFE_TensorHandle(TFE_Context* ctx, PyObject* obj,

```
....
740.              tensorflow::strings::StrCat("Invalid dtype argument
value ", dtype)
```

Dangerous Functions\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=22
Status	New

The dangerous function, realloc, was found in use at line 173 in tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Line	181	181
Object	realloc	realloc

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Method void TfLiteTensorRealloc(size_t num_bytes, TfLiteTensor* tensor) {

```
....
181.      tensor->data.raw = realloc(tensor->data.raw, num_bytes);
```

Dangerous Functions\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=23
Status	New

The dangerous function, realloc, was found in use at line 175 in tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Line	184	184
Object	realloc	realloc

Code Snippet

File Name	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Method	void TfLiteTensorRealloc(size_t num_bytes, TfLiteTensor* tensor) {

```
....
184.      tensor->data.raw = realloc(tensor->data.raw, num_bytes);
```

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-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=5
Status	New

The size of the buffer used by TfLiteIntArrayCopy in src, at line 52 of tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that TfLiteIntArrayCopy passes to src, at line 52 of tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c, to overwrite the target buffer.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Line	56	56
Object	src	src

Code Snippet

File Name	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Method	TfLiteIntArray* TfLiteIntArrayCopy(const TfLiteIntArray* src) {

```
....
56.      memcpy(ret->data, src->data, src->size * sizeof(int));
```

Buffer Overflow boundcpy WrongSizeParam\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=6
Status	New

The size of the buffer used by TfLiteIntArrayCopy in int, at line 52 of tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c, is not properly verified before writing data to the buffer. This can enable a buffer

overflow attack, using the source buffer that TfLiteIntArrayCopy passes to int, at line 52 of tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c, to overwrite the target buffer.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Line	56	56
Object	int	int

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Method TfLiteIntArray* TfLiteIntArrayCopy(const TfLiteIntArray* src) {

```
....
56.      memcpy(ret->data, src->data, src->size * sizeof(int));
```

Buffer Overflow boundcpy WrongSizeParam\Path 3:

Severity Medium
Result State To Verify
Online Results http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=7
Status New

The size of the buffer used by TfLiteIntArrayCopy in src, at line 52 of tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that TfLiteIntArrayCopy passes to src, at line 52 of tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c, to overwrite the target buffer.

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Line	56	56
Object	src	src

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Method TfLiteIntArray* TfLiteIntArrayCopy(const TfLiteIntArray* src) {

```
....
56.      memcpy(ret->data, src->data, src->size * sizeof(int));
```

Buffer Overflow boundcpy WrongSizeParam\Path 4:

Severity Medium
Result State To Verify
Online Results http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=8
Status New

The size of the buffer used by TfLiteIntArrayCopy in int, at line 52 of tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that TfLiteIntArrayCopy passes to int, at line 52 of tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c, to overwrite the target buffer.

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Line	56	56
Object	int	int

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Method TfLiteIntArray* TfLiteIntArrayCopy(const TfLiteIntArray* src) {

```
....
56.      memcpy(ret->data, src->data, src->size * sizeof(int));
```

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-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=24
Status New

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Line	73	73
Object	ret	ret

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Method TfLiteFloatArray* TfLiteFloatArrayCreate(int size) {

```
....
73.    TfLiteFloatArray* ret =
```

Memory Leak\Path 2:

Severity Medium
Result State To Verify

Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=25
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Line	73	73
Object	ret	ret

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Method TfLiteFloatArray* TfLiteFloatArrayCreate(int size) {

```
....
73.    TfLiteFloatArray* ret =
```

Memory Leak\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=26
Status	New

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Line	179	179
Object	raw	raw

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Method void TfLiteTensorRealloc(size_t num_bytes, TfLiteTensor* tensor) {

```
....
179.      tensor->data.raw = malloc(num_bytes);
```

Memory Leak\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=27
Status	New

	Source	Destination

File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Line	182	182
Object	raw	raw

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Method void TfLiteTensorRealloc(size_t num_bytes, TfLiteTensor* tensor) {

```
....
182.      tensor->data.raw = malloc(num_bytes);
```

MemoryFree on StackVariable

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

Description

MemoryFree on StackVariable\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=9
Status	New

Calling free() (line 88) on a variable that was not dynamically allocated (line 88) in file tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c may result with a crash.

	Source	Destination
File	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Line	100	100
Object	q_params	q_params

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Method void TfLiteQuantizationFree(TfLiteQuantization* quantization) {

```
....
100.      free(q_params);
```

MemoryFree on StackVariable\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=10
Status	New

Calling free() (line 89) on a variable that was not dynamically allocated (line 89) in file tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c may result with a crash.

	Source	Destination
File	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Line	101	101
Object	q_params	q_params

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Method void TfLiteQuantizationFree(TfLiteQuantization* quantization) {

```
....
101.      free(q_params);
```

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-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=1
Status	New

The TfLiteIntArrayCreate method calls the ret function, at line 45 of tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.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	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Line	46	46
Object	ret	ret

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Method TfLiteIntArray* TfLiteIntArrayCreate(int size) {

```
....
46.    TfLiteIntArray* ret =
```

Unchecked Return Value\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=2
Status	New

The TfLiteFloatArrayCreate method calls the ret function, at line 72 of tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.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	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Line	73	73
Object	ret	ret

Code Snippet
File Name tensorflow@@tensorflow-v2.2.0-rc1-CVE-2021-29605-FP.c
Method TfLiteFloatArray* TfLiteFloatArrayCreate(int size) {

```
....
73.    TfLiteFloatArray* ret =
```

Unchecked Return Value\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=3
Status	New

The TfLiteIntArrayCreate method calls the ret function, at line 45 of tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.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	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Line	46	46
Object	ret	ret

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Method TfLiteIntArray* TfLiteIntArrayCreate(int size) {

```
....
46.    TfLiteIntArray* ret =
```

Unchecked Return Value\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-PTJMSNK3USL/CxWebClient/ViewerMain.aspx?scanid=1030070&projectid=30059&pathid=4
Status	New

The TfLiteFloatArrayCreate method calls the ret function, at line 72 of tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.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	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c	tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Line	73	73
Object	ret	ret

Code Snippet
File Name tensorflow@@tensorflow-v2.3.0-rc0-CVE-2021-29605-FP.c
Method TfLiteFloatArray* TfLiteFloatArrayCreate(int size) {

```
....
73.    TfLiteFloatArray* ret =
```

Buffer Overflow boundcpy WrongSizeParam

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

- Always perform proper bounds checking before copying buffers or strings.

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

Source Code Examples

CPP

Overflowing Buffers

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

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

Checked Buffers

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

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

MemoryFree on StackVariable

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

Source Code Examples

CPP

Bad - Calling free() on a static variable

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

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

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

Dangerous Functions

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

Source Code Examples

CPP

Buffer Overflow in gets()

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

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

Safe reading from user

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

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

Unsafe function for string copy

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

	return 0;
}
```

Safe string copy

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

	return 0;
}
```

Unsafe format string

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

Safe format string

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

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

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

Description

Description Summary

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

Extended Description

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

Terminology Notes

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

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

C

C++

Modes of Introduction

Memory leaks have two common and sometimes overlapping causes:

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

Common Consequences

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

Likelihood of Exploit

Medium

Demonstrative Examples

Example 1

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

(Bad Code)

Example Language: **C**

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

return NULL;
}
```

```
return buf;
}
```

Example 2

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

(Bad Code)

Example Language: **C**

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

free(foo);
}
int main() {

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

Observed Examples

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

Potential Mitigations

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

Phase: Architecture and Design

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

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

Relationships

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

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

Relationship Notes

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

Affected Resources

- Memory

Functional Areas

- Memory management

Taxonomy Mappings

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

White Box Definitions

A weakness where the code path has:

1. start statement that allocates dynamically allocated memory resource

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

Where "loses" is defined through the following scenarios:

1. identity of the dynamic allocated memory resource never obtained

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

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

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

References

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

Content History

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

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

Previous Entry Names

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

BACK TO TOP

Use of Uninitialized Variable

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

Description

Description Summary

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

Extended Description

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

Time of Introduction

- Implementation

Applicable Platforms

Languages

C: *(Sometimes)*

C++: *(Sometimes)*

Perl: *(Often)*

All

Common Consequences

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

Likelihood of Exploit

High

Demonstrative Examples

Example 1

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

(Bad Code)

Example Language: **C**

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

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

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

Example 2

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

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

Observed Examples

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

Potential Mitigations

Phase: Implementation

Assign all variables to an initial value.

Phase: Build and Compilation

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

Phase: Requirements

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

Phase: Architecture and Design

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

Other Notes

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

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

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	398	Indicator of Poor Code Quality	**Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Base	456	Missing Initialization	**Development Concepts (primary)699 Research Concepts**

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

Taxonomy Mappings

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

White Box Definitions

A weakness where the code path has:

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

References

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

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

Content History

Submissions

Submission Date	Submitter	Organization	Source
	CLASP		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci	Cigital	External
	updated Time of Introduction		
2008-08-01		KDM Analytics	External
	added/updated white box definitions		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Common Consequences, Description, Relationships, Observed Example, Other Notes, References, Taxonomy Mappings		
2009-01-12	CWE Content Team	MITRE	Internal
	updated Common Consequences, Demonstrative Examples, Potential Mitigations		
2009-03-10	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples		

Previous Entry Names

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

BACK TO TOP

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';
```

Scanned Languages

Language	Hash Number	Change Date
CPP	4541647240435660	1/6/2025
Common	0105849645654507	1/6/2025