emscripten Scan Report

Project Name	emscripten
Scan Start	Friday, June 21, 2024 11:19:18 PM
Preset	Checkmarx Default
Scan Time	00h:13m:41s
Lines Of Code Scanned	27058
Files Scanned	42
Report Creation Time	Friday, June 21, 2024 11:38:44 PM
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064
Team	CxServer
Checkmarx Version	8.7.0
Scan Type	Full
Source Origin	LocalPath
Density	8/1000 (Vulnerabilities/LOC)
Visibility	Public

Filter Settings

Severity

Included: High, Medium, Low, Information

Excluded: None

Result State

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

Excluded: None

Assigned to

Included: All

Categories

Included:

Uncategorized	All
Custom	All
PCI DSS v3.2	All
OWASP Top 10 2013	All
FISMA 2014	All
NIST SP 800-53	All
OWASP Top 10 2017	All
OWASP Mobile Top 10 2016	All

Excluded:

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

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

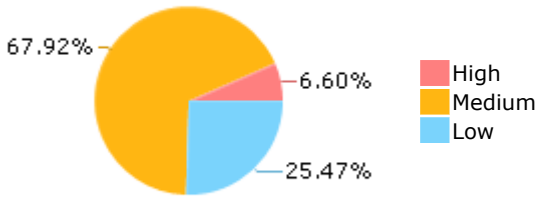
Results Limit

Results limit per query was set to 50

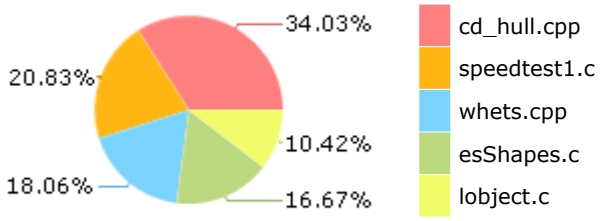
Selected Queries

Selected queries are listed in Result Summary

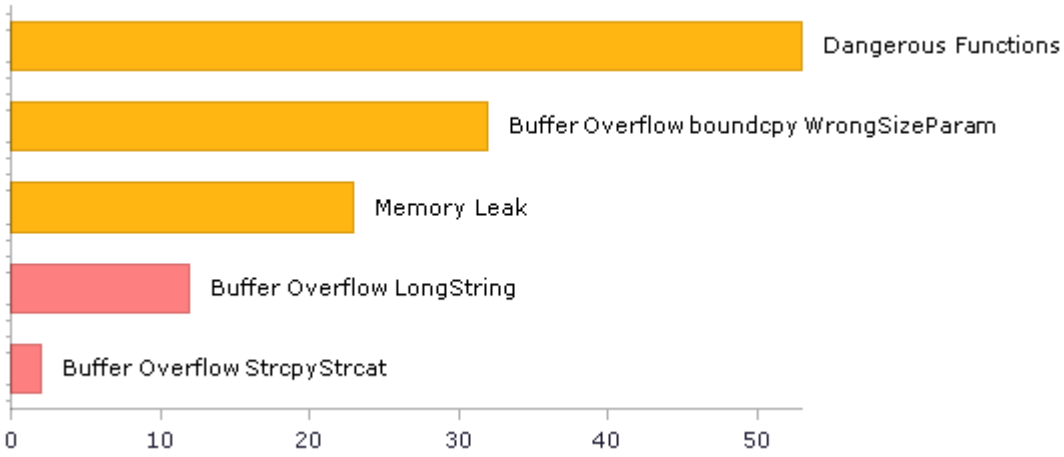
Result Summary



Most Vulnerable Files



Top 5 Vulnerabilities



Scan Summary - OWASP Top 10 2017

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

Category	Threat Agent	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impact	Business Impact	Issues Found	Best Fix Locations
A1-Injection	App. Specific	EASY	COMMON	EASY	SEVERE	App. Specific	53	42
A2-Broken Authentication	App. Specific	EASY	COMMON	AVERAGE	SEVERE	App. Specific	1	1
A3-Sensitive Data Exposure	App. Specific	AVERAGE	WIDESPREAD	AVERAGE	SEVERE	App. Specific	1	1
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	53	53
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	53	53
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	2	2
PCI DSS (3.2) - 6.5.2 - Buffer overflows	69	58
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.	1	1
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.	1	1
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.	15	15

***** 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)	1	1
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)	1	1
SC-4 Information in Shared Resources (P1)	0	0
SC-5 Denial of Service Protection (P1)*	25	25
SC-8 Transmission Confidentiality and Integrity (P1)	0	0
SI-10 Information Input Validation (P1)*	54	43
SI-11 Error Handling (P2)*	24	24
SI-15 Information Output Filtering (P0)	0	0
SI-16 Memory Protection (P1)	2	2

***** 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

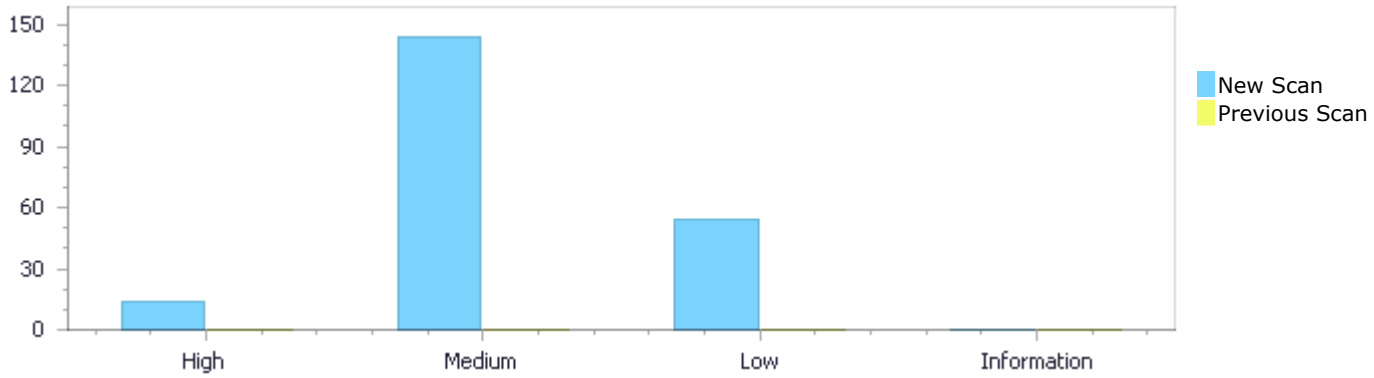
Scan Summary - Custom

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	14	144	54	0	212
Recurrent Issues	0	0	0	0	0
Total	14	144	54	0	212
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	14	144	54	0	212
Urgent	0	0	0	0	0
Proposed Not Exploitable	0	0	0	0	0
Total	14	144	54	0	212

Result Summary

Vulnerability Type	Occurrences	Severity
Buffer Overflow LongString	12	High
Buffer Overflow StrcpyStrcat	2	High
Dangerous Functions	53	Medium
Buffer Overflow boundcpy WrongSizeParam	32	Medium
Memory Leak	23	Medium

Short Overflow	9	Medium
Char Overflow	6	Medium
Divide By Zero	6	Medium
MemoryFree on StackVariable	6	Medium
Integer Overflow	5	Medium
Buffer Overflow AddressOfLocalVarReturned	1	Medium
Long Overflow	1	Medium
Use After Free	1	Medium
Wrong Size t Allocation	1	Medium
Unchecked Return Value	24	Low
Unchecked Array Index	16	Low
Sizeof Pointer Argument	3	Low
Use of Sizeof On a Pointer Type	3	Low
Potential Off by One Error in Loops	2	Low
Potential Precision Problem	2	Low
Heuristic Buffer Overflow malloc	1	Low
Improper Resource Access Authorization	1	Low
Inconsistent Implementations	1	Low
Use of Insufficiently Random Values	1	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
emscripten/cd_hull.cpp	46
emscripten/speedtest1.c	22
emscripten/esShapes.c	16
emscripten/whets.cpp	15
emscripten/lobject.c	14
emscripten/crypt_blowfish.c	10
emscripten/crypt_sha256.c	6
emscripten/linpack2.c	5
emscripten/lvm.c	4
emscripten/__secs_to_tm.c	4

Scan Results Details

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=1050074&projectid=50064&pathid=1
Status	New

The size of the buffer used by BF_set_key in tmp, at line 498 of emscripten/crypt_blowfish.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *__crypt_blowfish passes to "8b \xd0\xc1\xd2\xcf\xcc\xd8", at line 748 of emscripten/crypt_blowfish.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c
Line	750	549
Object	"8b \xd0\xc1\xd2\xcf\xcc\xd8"	tmp

Code Snippet
File Name emscripten/crypt_blowfish.c
Method char *__crypt_blowfish(const char *key, const char *setting, char *output)

```
....
750.        const char *test_key = "8b \xd0\xc1\xd2\xcf\xcc\xd8";
```

File Name emscripten/crypt_blowfish.c
Method static void BF_set_key(const char *key, BF_key expanded, BF_key initial,

```
....
549.                    tmp[0] |= (unsigned char)*ptr; /* correct */
```

Buffer Overflow LongString\Path 2:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=2

Status	New

The size of the buffer used by BF_set_key in tmp, at line 498 of emscripten/crypt_blowfish.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *__crypt_blowfish passes to "8b \xd0\xc1\xd2\xcf\xcc\xd8", at line 748 of emscripten/crypt_blowfish.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c
Line	750	551
Object	"8b \xd0\xc1\xd2\xcf\xcc\xd8"	tmp

Code Snippet
File Name emscripten/crypt_blowfish.c
Method char *__crypt_blowfish(const char *key, const char *setting, char *output)

```
....
750.        const char *test_key = "8b \xd0\xc1\xd2\xcf\xcc\xd8";
```

File Name emscripten/crypt_blowfish.c
Method static void BF_set_key(const char *key, BF_key expanded, BF_key initial,

```
....
551.                    tmp[1] |= (signed char)*ptr; /* bug */
```

Buffer Overflow LongString\Path 3:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=3
Status	New

The size of the buffer used by BF_set_key in tmp, at line 498 of emscripten/crypt_blowfish.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *__crypt_blowfish passes to "\xff\xa3", at line 748 of emscripten/crypt_blowfish.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c
Line	792	549
Object	"\xff\xa3"	tmp

Code Snippet
File Name emscripten/crypt_blowfish.c
Method char *__crypt_blowfish(const char *key, const char *setting, char *output)

```
....
792.              const char *k = "\xff\xa3" "34" "\xff\xff\xff\xa3"
"345";
```

File Name emscripten/crypt_blowfish.c

Method static void BF_set_key(const char *key, BF_key expanded, BF_key initial,

```
....
549.                    tmp[0] |= (unsigned char)*ptr; /* correct */
```

Buffer Overflow LongString\Path 4:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=4
Status	New

The size of the buffer used by BF_set_key in tmp, at line 498 of emscripten/crypt_blowfish.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *__crypt_blowfish passes to "\xff\xa3", at line 748 of emscripten/crypt_blowfish.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c
Line	792	551
Object	"\xff\xa3"	tmp

Code Snippet

File Name emscripten/crypt_blowfish.c

Method char *__crypt_blowfish(const char *key, const char *setting, char *output)

```
....
792.              const char *k = "\xff\xa3" "34" "\xff\xff\xff\xa3"
"345";
```

File Name emscripten/crypt_blowfish.c

Method static void BF_set_key(const char *key, BF_key expanded, BF_key initial,

```
....
551.                    tmp[1] |= (signed char)*ptr; /* bug */
```

Buffer Overflow LongString\Path 5:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=500

	64&pathid=5
Status	New

The size of the buffer used by pout in headings, at line 575 of emscripten/whets.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that whetstones passes to "N1 floating point\0", at line 345 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	393	591
Object	"N1 floating point\0"	headings

Code Snippet
File Name emscripten/whets.cpp
Method void whetstones(long xtra, long x100, int calibrate)

```
....
393.        pout("N1 floating point\0",(float)(n1*16)*(float)(xtra),
```

File Name emscripten/whets.cpp
Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
591.              printf("%s %24.17f
",headings[section],results[section]);
```

Buffer Overflow LongString\Path 6:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=6
Status	New

The size of the buffer used by pout in headings, at line 575 of emscripten/whets.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that whetstones passes to "N2 floating point\0", at line 345 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	411	591
Object	"N2 floating point\0"	headings

Code Snippet
File Name emscripten/whets.cpp
Method void whetstones(long xtra, long x100, int calibrate)

```
....
411.        pout("N2 floating point\0",(float)(n2*96)*(float)(xtra),
```

File Name emscripten/whets.cpp

Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
591.              printf("%s %24.17f
",headings[section],results[section]);
```

Buffer Overflow LongString\Path 7:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=7
Status	New

The size of the buffer used by pout in headings, at line 575 of emscripten/whets.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that whetstones passes to "N3 if then else \0", at line 345 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	432	591
Object	"N3 if then else \0"	headings

Code Snippet

File Name emscripten/whets.cpp

Method void whetstones(long xtra, long x100, int calibrate)

```
....
432.        pout("N3 if then else  \0",(float)(n3*3)*(float)(xtra),
```

File Name emscripten/whets.cpp

Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
591.              printf("%s %24.17f
",headings[section],results[section]);
```

Buffer Overflow LongString\Path 8:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=8

Status	New

The size of the buffer used by pout in headings, at line 575 of emscripten/whets.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that whetstones passes to "N4 fixed point \0", at line 345 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	458	591
Object	"N4 fixed point \0"	headings

Code Snippet
File Name emscripten/whets.cpp
Method void whetstones(long xtra, long x100, int calibrate)

```
....
458.        pout("N4 fixed point   \0",(float)(n4*15)*(float)(xtra),
```

File Name emscripten/whets.cpp
Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
591.              printf("%s %24.17f
",headings[section],results[section]);
```

Buffer Overflow LongString\Path 9:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=9
Status	New

The size of the buffer used by pout in headings, at line 575 of emscripten/whets.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that whetstones passes to "N5 sin,cos etc. \0", at line 345 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	478	591
Object	"N5 sin,cos etc. \0"	headings

Code Snippet
File Name emscripten/whets.cpp
Method void whetstones(long xtra, long x100, int calibrate)

```
....
478.        pout("N5 sin,cos etc.  \0",(float)(n5*26)*(float)(xtra),
```

File Name emscripten/whets.cpp

Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
591.              printf("%s %24.17f
",headings[section],results[section]);
```

Buffer Overflow LongString\Path 10:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=10
Status	New

The size of the buffer used by pout in headings, at line 575 of emscripten/whets.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that whetstones passes to "N6 floating point\0", at line 345 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	498	591
Object	"N6 floating point\0"	headings

Code Snippet

File Name emscripten/whets.cpp

Method void whetstones(long xtra, long x100, int calibrate)

```
....
498.        pout("N6 floating point\0",(float)(n6*6)*(float)(xtra),
```

File Name emscripten/whets.cpp

Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
591.              printf("%s %24.17f
",headings[section],results[section]);
```

Buffer Overflow LongString\Path 11:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=11
Status	New

The size of the buffer used by pout in headings, at line 575 of emscripten/whets.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that whetstones passes to "N7 assignments \0", at line 345 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	519	591
Object	"N7 assignments \0"	headings

Code Snippet
File Name emscripten/whets.cpp
Method void whetstones(long xtra, long x100, int calibrate)

```
....
519.        pout("N7 assignments    \0",(float)(n7*3)*(float)(xtra),
```

File Name emscripten/whets.cpp
Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
591.              printf("%s %24.17f
",headings[section],results[section]);
```

Buffer Overflow LongString\Path 12:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=12
Status	New

The size of the buffer used by pout in headings, at line 575 of emscripten/whets.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that whetstones passes to "N8 exp,sqrt etc. \0", at line 345 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	535	591
Object	"N8 exp,sqrt etc. \0"	headings

Code Snippet
File Name emscripten/whets.cpp
Method void whetstones(long xtra, long x100, int calibrate)

```
....
535.        pout("N8 exp,sqrt etc. \0",(float)(n8*4)*(float)(xtra),
```

File Name emscripten/whets.cpp

Method	void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
591.              printf("%s %24.17f
",headings[section],results[section]);
```

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=1050074&projectid=50064&pathid=90
Status	New

The size of the buffer used by pout in title, at line 575 of emscripten/whets.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pout passes to title, at line 575 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	575	582
Object	title	title

Code Snippet	
File Name	emscripten/whets.cpp
Method	void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
575.  void pout(const char title[18], float ops, int type, SPDP
checknum,
....
582.        strcpy (headings[section],title);
```

Buffer Overflow StrcpyStrcat\Path 2:

Severity	High
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=91
Status	New

The size of the buffer used by pout in headings, at line 575 of emscripten/whets.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that pout passes to title, at line 575 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	575	582
Object	title	headings

Code Snippet
File Name emscripten/whets.cpp
Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
575.  void pout(const char title[18], float ops, int type, SPDP
checknum,
....
582.        strcpy (headings[section],title);
```

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=1050074&projectid=50064&pathid=135
Status	New

The dangerous function, memcpy, was found in use at line 748 in emscripten/crypt_blowfish.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c
Line	775	775
Object	memcpy	memcpy

Code Snippet
File Name emscripten/crypt_blowfish.c
Method char *__crypt_blowfish(const char *key, const char *setting, char *output)

```
....
775.        memcpy(buf.s, test_setting, sizeof(buf.s));
```

Dangerous Functions\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=136
Status	New

The dangerous function, memcpy, was found in use at line 605 in emscripten/crypt_blowfish.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c
Line	639	639
Object	memcpy	memcpy

Code Snippet
File Name emscripten/crypt_blowfish.c
Method static char *BF_crypt(const char *key, const char *setting,

```
....
639.        memcpy(data.ctx.s.S, BF_init_state.s.S,
sizeof(data.ctx.s.S));
```

Dangerous Functions\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=137
Status	New

The dangerous function, memcpy, was found in use at line 605 in emscripten/crypt_blowfish.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c
Line	715	715
Object	memcpy	memcpy

Code Snippet
File Name emscripten/crypt_blowfish.c
Method static char *BF_crypt(const char *key, const char *setting,

```
....
715.        memcpy(output, setting, 7 + 22 - 1);
```

Dangerous Functions\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=138
Status	New

The dangerous function, memcpy, was found in use at line 135 in emscripten/crypt_sha256.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/crypt_sha256.c	emscripten/crypt_sha256.c
Line	143	143
Object	memcpy	memcpy

Code Snippet
File Name emscripten/crypt_sha256.c
Method static void sha256_update(struct sha256 *s, const void *m, unsigned long len)

```
....
143.                    memcpy(s->buf + r, p, len);
```

Dangerous Functions\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=139
Status	New

The dangerous function, memcpy, was found in use at line 135 in emscripten/crypt_sha256.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/crypt_sha256.c	emscripten/crypt_sha256.c
Line	146	146
Object	memcpy	memcpy

Code Snippet
File Name emscripten/crypt_sha256.c
Method static void sha256_update(struct sha256 *s, const void *m, unsigned long len)

```
....
146.                memcpy(s->buf + r, p, 64 - r);
```

Dangerous Functions\Path 6:

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

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=140
Status	New

The dangerous function, memcpy, was found in use at line 135 in emscripten/crypt_sha256.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/crypt_sha256.c	emscripten/crypt_sha256.c
Line	153	153
Object	memcpy	memcpy

Code Snippet
File Name emscripten/crypt_sha256.c
Method static void sha256_update(struct sha256 *s, const void *m, unsigned long len)

```
....
153.        memcpy(s->buf, p, len);
```

Dangerous Functions\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=141
Status	New

The dangerous function, memcpy, was found in use at line 141 in emscripten/esShapes.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	236	236
Object	memcpy	memcpy

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
236.        memcpy( *vertices, cubeVerts, sizeof( cubeVerts ) );
```

Dangerous Functions\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=142
Status	New

The dangerous function, memcpy, was found in use at line 141 in emscripten/esShapes.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	246	246
Object	memcpy	memcpy

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
246.        memcpy( *normals, cubeNormals, sizeof( cubeNormals ) );
```

Dangerous Functions\Path 9:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=143
Status New

The dangerous function, memcpy, was found in use at line 141 in emscripten/esShapes.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	252	252
Object	memcpy	memcpy

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
252.        memcpy( *texCoords, cubeTex, sizeof( cubeTex ) ) ;
```

Dangerous Functions\Path 10:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=144
Status New

The dangerous function, memcpy, was found in use at line 141 in emscripten/esShapes.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	276	276
Object	memcpy	memcpy

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
276.        memcpy( *indices, cubeIndices, sizeof( cubeIndices ) );
```

Dangerous Functions\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=145
Status	New

The dangerous function, memcpy, was found in use at line 252 in emscripten/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	256	256
Object	memcpy	memcpy

Code Snippet
File Name emscripten/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
256.        memcpy(out, source + 1, l * sizeof(char));
```

Dangerous Functions\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=146
Status	New

The dangerous function, memcpy, was found in use at line 252 in emscripten/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c

Line	264	264
Object	memcpy	memcpy

Code Snippet
File Name emscripten/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
264.        memcpy(out, source + 1, l * sizeof(char));
```

Dangerous Functions\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=147
Status	New

The dangerous function, memcpy, was found in use at line 252 in emscripten/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	268	268
Object	memcpy	memcpy

Code Snippet
File Name emscripten/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
268.        memcpy(out, source + 1 + l - bufflen, bufflen *
sizeof(char));
```

Dangerous Functions\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=148
Status	New

The dangerous function, memcpy, was found in use at line 252 in emscripten/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	284	284

Object	memcpy	memcpy

Code Snippet
File Name emscripten/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
284.      memcpy(out, POS, (LL(POS) + 1) * sizeof(char));
```

Dangerous Functions\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=149
Status	New

The dangerous function, memcpy, was found in use at line 293 in emscripten/lvm.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/lvm.c	emscripten/lvm.c
Line	324	324
Object	memcpy	memcpy

Code Snippet
File Name emscripten/lvm.c
Method void luaV_concat (lua_State *L, int total) {

```
....
324.          memcpy(buffer+tl, svalue(top-i), l * sizeof(char));
```

Dangerous Functions\Path 16:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=150
Status	New

The dangerous function, memcpy, was found in use at line 361 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	374	374
Object	memcpy	memcpy

Code Snippet	
File Name	emscripten/speedtest1.c
Method	void speedtest1_execute(void){

```
....
374.          memcpy(g.zResult + g.nResult, z, len+1);
```

Dangerous Functions\Path 17:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=151
Status	New

The dangerous function, memcpy, was found in use at line 1666 in emscripten/cd_hull.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1668	1668
Object	memcpy	memcpy

Code Snippet	
File Name	emscripten/cd_hull.cpp
Method	ConvexH *ConvexHDup(ConvexH *src) {

```
....
1668.       memcpy(dst->vertices.element,src-
>vertices.element,sizeof(float3)*src->vertices.count);
```

Dangerous Functions\Path 18:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=152
Status	New

The dangerous function, memcpy, was found in use at line 1666 in emscripten/cd_hull.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1669	1669
Object	memcpy	memcpy

Code Snippet	
File Name	emscripten/cd_hull.cpp

Method	ConvexH *ConvexHDup(ConvexH *src) {

```
....
1669.       memcpy(dst->edges.element,src-
>edges.element,sizeof(HalfEdge)*src->edges.count);
```

Dangerous Functions\Path 19:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=153
Status	New

The dangerous function, memcpy, was found in use at line 1666 in emscripten/cd_hull.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1670	1670
Object	memcpy	memcpy

Code Snippet	
File Name	emscripten/cd_hull.cpp
Method	ConvexH *ConvexHDup(ConvexH *src) {

```
....
1670.       memcpy(dst->facets.element,src-
>facets.element,sizeof(Plane)*src->facets.count);
```

Dangerous Functions\Path 20:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=154
Status	New

The dangerous function, memcpy, was found in use at line 1851 in emscripten/cd_hull.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2146	2146
Object	memcpy	memcpy

Code Snippet	
File Name	emscripten/cd_hull.cpp
Method	ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2146.
      memcpy(under.edges.element,tmpunderedges,sizeof(HalfEdge)*under_ed
ge_count);
```

Dangerous Functions\Path 21:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=155
Status	New

The dangerous function, memcpy, was found in use at line 1851 in emscripten/cd_hull.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2147	2147
Object	memcpy	memcpy

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2147.
      memcpy(under.facets.element,tmpunderplanes,sizeof(Plane)*underplan
escount);
```

Dangerous Functions\Path 22:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=156
Status	New

The dangerous function, memcpy, was found in use at line 2792 in emscripten/cd_hull.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2853	2853
Object	memcpy	memcpy

Code Snippet
File Name emscripten/cd_hull.cpp

Method	HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2853.                          memcpy(result.mOutputVertices, vscratch,
sizeof(float)*3*ovcount );
```

Dangerous Functions\Path 23:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=157
Status	New

The dangerous function, memcpy, was found in use at line 2792 in emscripten/cd_hull.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2873	2873
Object	memcpy	memcpy

Code Snippet	
File Name	emscripten/cd_hull.cpp
Method	HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2873.                              memcpy(result.mIndices, hr.mIndices,
sizeof(unsigned int)*hr.mIndexCount);
```

Dangerous Functions\Path 24:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=158
Status	New

The dangerous function, memcpy, was found in use at line 2792 in emscripten/cd_hull.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2884	2884
Object	memcpy	memcpy

Code Snippet

File Name	emscripten/cd_hull.cpp
Method	HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2884.                           memcpy(result.mOutputVertices, vscratch,
sizeof(float)*3*ovcount );
```

Dangerous Functions\Path 25:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=159
Status	New

The dangerous function, sprintf, was found in use at line 187 in emscripten/crypt_sha256.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/crypt_sha256.c	emscripten/crypt_sha256.c
Line	237	237
Object	sprintf	sprintf

Code Snippet	
File Name	emscripten/crypt_sha256.c
Method	static char *sha256crypt(const char *key, const char *setting, char *output)

```
....
237.              sprintf(rounds, "rounds=%u$", r);
```

Dangerous Functions\Path 26:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=160
Status	New

The dangerous function, sprintf, was found in use at line 187 in emscripten/crypt_sha256.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/crypt_sha256.c	emscripten/crypt_sha256.c
Line	297	297
Object	sprintf	sprintf

Code Snippet	
File Name	emscripten/crypt_sha256.c

Method	static char *sha256crypt(const char *key, const char *setting, char *output)

```
....
297.        p += sprintf(p, "$5$%s%.*s$", rounds, slen, salt);
```

Dangerous Functions\Path 27:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=161
Status	New

The dangerous function, sprintf, was found in use at line 256 in emscripten/whets.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	273	273
Object	sprintf	sprintf

Code Snippet	
File Name	emscripten/whets.cpp
Method	int main(int argc, char *argv[])

```
....
273.              sprintf(timeday, "%s", asctime(localtime(&t)));
```

Dangerous Functions\Path 28:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=162
Status	New

The dangerous function, strcpy, was found in use at line 575 in emscripten/whets.cpp file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	582	582
Object	strcpy	strcpy

Code Snippet	
File Name	emscripten/whets.cpp
Method	void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
582.        strcpy (headings[section],title);
```

Dangerous Functions\Path 29:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=163
Status	New

The dangerous function, strlen, was found in use at line 252 in emscripten/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	253	253
Object	strlen	strlen

Code Snippet
File Name emscripten/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
253.    size_t l = strlen(source);
```

Dangerous Functions\Path 30:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=164
Status	New

The dangerous function, strlen, was found in use at line 179 in emscripten/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	190	190
Object	strlen	strlen

Code Snippet
File Name emscripten/lobject.c
Method const char *luaO_pushvfstring (lua_State *L, const char *fmt, va_list argp) {

```
....
190.          pushstr(L, s, strlen(s));
```

Dangerous Functions\Path 31:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=165
Status	New

The dangerous function, strlen, was found in use at line 179 in emscripten/lobject.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	227	227
Object	strlen	strlen

Code Snippet
File Name emscripten/lobject.c
Method const char *luaO_pushvfstring (lua_State *L, const char *fmt, va_list argp) {

```
....
227.    pushstr(L, fmt, strlen(fmt));
```

Dangerous Functions\Path 32:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=166
Status	New

The dangerous function, strlen, was found in use at line 209 in emscripten/lvm.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/lvm.c	emscripten/lvm.c
Line	218	218
Object	strlen	strlen

Code Snippet
File Name emscripten/lvm.c
Method static int l_strcmp (const TString *ls, const TString *rs) {

```
....
218.        size_t len = strlen(l);  /* index of first `\0' in both
strings */
```

Dangerous Functions\Path 33:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=167
Status	New

The dangerous function, strlen, was found in use at line 200 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	212	212
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method int speedtest1_numbername(unsigned int n, char *zOut, int nOut){

```
....
212.      i += (int)strlen(zOut+i);
```

Dangerous Functions\Path 34:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=168
Status	New

The dangerous function, strlen, was found in use at line 200 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	219	219
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method int speedtest1_numbername(unsigned int n, char *zOut, int nOut){

```
....
219.        i += (int)strlen(zOut+i);
```

Dangerous Functions\Path 35:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=169
Status	New

The dangerous function, strlen, was found in use at line 200 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	226	226
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method int speedtest1_numbername(unsigned int n, char *zOut, int nOut){

```
....
226.        i += (int)strlen(zOut+i);
```

Dangerous Functions\Path 36:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=170
Status	New

The dangerous function, strlen, was found in use at line 200 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	232	232
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method int speedtest1_numbername(unsigned int n, char *zOut, int nOut){

```
....
232.        i += (int)strlen(zOut+i);
```

Dangerous Functions\Path 37:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=171
Status	New

The dangerous function, strlen, was found in use at line 200 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	238	238
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method int speedtest1_numbername(unsigned int n, char *zOut, int nOut){

```
....
238.        i += (int)strlen(zOut+i);
```

Dangerous Functions\Path 38:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=172
Status	New

The dangerous function, strlen, was found in use at line 200 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	244	244
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method int speedtest1_numbername(unsigned int n, char *zOut, int nOut){

```
....
244.        i += (int)strlen(zOut+i);
```

Dangerous Functions\Path 39:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=173
Status	New

The dangerous function, strlen, was found in use at line 200 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	248	248
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method int speedtest1_numbername(unsigned int n, char *zOut, int nOut){

```
....
248.        i += (int)strlen(zOut+i);
```

Dangerous Functions\Path 40:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=174
Status	New

The dangerous function, strlen, was found in use at line 258 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	259	259
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method void speedtest1_begin_test(int iTestNum, const char *zTestName, ...){

```
....
259.    int n = (int)strlen(zTestName);
```

Dangerous Functions\Path 41:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=175
Status	New

The dangerous function, strlen, was found in use at line 258 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	265	265
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method void speedtest1_begin_test(int iTestNum, const char *zTestName, ...){

```
....
265.    n = (int)strlen(zName);
```

Dangerous Functions\Path 42:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=176
Status	New

The dangerous function, strlen, was found in use at line 305 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	306	306
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method static void printSql(const char *zSql){

```
....
306.    int n = (int)strlen(zSql);
```

Dangerous Functions\Path 43:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=177
Status	New

The dangerous function, strlen, was found in use at line 361 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	371	371
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method void speedtest1_execute(void){

```
....
371.        len = (int)strlen(z);
```

Dangerous Functions\Path 44:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=178
Status	New

The dangerous function, strlen, was found in use at line 390 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	391	391
Object	strlen	strlen

Code Snippet
File Name emscripten/speedtest1.c
Method static void traceCallback(void *NotUsed, const char *zSql){

```
....
391.    int n = (int)strlen(zSql);
```

Dangerous Functions\Path 45:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=179
Status	New

The dangerous function, atoi, was found in use at line 989 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	1032	1032
Object	atoi	atoi

Code Snippet
File Name emscripten/speedtest1.c
Method void testset_rtree(int p1, int p2){

```
....
1032.       aCheck[i] = atoi(g.zResult);
```

Dangerous Functions\Path 46:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=180
Status	New

The dangerous function, atoi, was found in use at line 989 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	1045	1045
Object	atoi	atoi

Code Snippet
File Name emscripten/speedtest1.c
Method void testset_rtree(int p1, int p2){

```
....
1045.        if( aCheck[i]!=atoi(g.zResult) ){
```

Dangerous Functions\Path 47:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=181
Status	New

The dangerous function, atoi, was found in use at line 989 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	1047	1047
Object	atoi	atoi

Code Snippet
File Name emscripten/speedtest1.c
Method void testset_rtree(int p1, int p2){

```
....
1047.                       i, i*iStep, (i+1)*iStep, aCheck[i],
atoi(g.zResult));
```

Dangerous Functions\Path 48:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=182
Status	New

The dangerous function, atoi, was found in use at line 989 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	1061	1061
Object	atoi	atoi

Code Snippet
File Name emscripten/speedtest1.c
Method void testset_rtree(int p1, int p2){

```
....
1061.       aCheck[i] = atoi(g.zResult);
```

Dangerous Functions\Path 49:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=183
Status	New

The dangerous function, atoi, was found in use at line 989 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	1074	1074
Object	atoi	atoi

Code Snippet
File Name emscripten/speedtest1.c
Method void testset_rtree(int p1, int p2){

```
....
1074.         if( aCheck[i]!=atoi(g.zResult) ){
```

Dangerous Functions\Path 50:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=184
Status	New

The dangerous function, atoi, was found in use at line 989 in emscripten/speedtest1.c file. Such functions may expose information and allow an attacker to get full control over the host machine.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	1076	1076
Object	atoi	atoi

Code Snippet
File Name emscripten/speedtest1.c
Method void testset_rtree(int p1, int p2){

```
....
1076.                       i, i*iStep, (i+1)*iStep, aCheck[i],
atoi(g.zResult));
```

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=1050074&projectid=50064&pathid=51
Status	New

The size of the buffer used by *__crypt_blowfish in Namespace765909512, at line 748 of emscripten/crypt_blowfish.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *__crypt_blowfish passes to Namespace765909512, at line 748 of emscripten/crypt_blowfish.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c
Line	775	775
Object	Namespace765909512	Namespace765909512

Code Snippet
File Name emscripten/crypt_blowfish.c
Method char *__crypt_blowfish(const char *key, const char *setting, char *output)

```
....
775.        memcpy(buf.s, test_setting, sizeof(buf.s));
```

Buffer Overflow boundcpy WrongSizeParam\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=52
Status	New

The size of the buffer used by *BF_crypt in Namespace765909512, at line 605 of emscripten/crypt_blowfish.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *BF_crypt passes to Namespace765909512, at line 605 of emscripten/crypt_blowfish.c, to overwrite the target buffer.

	Source	Destination

File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c
Line	639	639
Object	Namespace765909512	Namespace765909512

Code Snippet
File Name emscripten/crypt_blowfish.c
Method static char *BF_crypt(const char *key, const char *setting,

```
....
639.        memcpy(data.ctx.s.S, BF_init_state.s.S,
sizeof(data.ctx.s.S));
```

Buffer Overflow boundcpy WrongSizeParam\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=53
Status	New

The size of the buffer used by esGenCube in cubeVerts, at line 141 of emscripten/esShapes.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that esGenCube passes to cubeVerts, at line 141 of emscripten/esShapes.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	236	236
Object	cubeVerts	cubeVerts

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
236.        memcpy( *vertices, cubeVerts, sizeof( cubeVerts ) );
```

Buffer Overflow boundcpy WrongSizeParam\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=54
Status	New

The size of the buffer used by esGenCube in cubeNormals, at line 141 of emscripten/esShapes.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that esGenCube passes to cubeNormals, at line 141 of emscripten/esShapes.c, to overwrite the target buffer.

	Source	Destination

File	emscripten/esShapes.c	emscripten/esShapes.c
Line	246	246
Object	cubeNormals	cubeNormals

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
246.        memcpy( *normals, cubeNormals, sizeof( cubeNormals ) );
```

Buffer Overflow boundcpy WrongSizeParam\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=55
Status	New

The size of the buffer used by esGenCube in cubeTex, at line 141 of emscripten/esShapes.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that esGenCube passes to cubeTex, at line 141 of emscripten/esShapes.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	252	252
Object	cubeTex	cubeTex

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
252.        memcpy( *texCoords, cubeTex, sizeof( cubeTex ) ) ;
```

Buffer Overflow boundcpy WrongSizeParam\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=56
Status	New

The size of the buffer used by esGenCube in cubeIndices, at line 141 of emscripten/esShapes.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that esGenCube passes to cubeIndices, at line 141 of emscripten/esShapes.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c

Line	276	276
Object	cubeIndices	cubeIndices

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
276.        memcpy( *indices, cubeIndices, sizeof( cubeIndices ) );
```

Buffer Overflow boundcpy WrongSizeParam\Path 7:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=57
Status	New

The size of the buffer used by *__crypt_blowfish in Namespace765909512, at line 748 of emscripten/crypt_blowfish.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *__crypt_blowfish passes to Namespace765909512, at line 748 of emscripten/crypt_blowfish.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c
Line	781	781
Object	Namespace765909512	Namespace765909512

Code Snippet
File Name emscripten/crypt_blowfish.c
Method char *__crypt_blowfish(const char *key, const char *setting, char *output)

```
....
781.        memset(buf.o, 0x55, sizeof(buf.o));
```

Buffer Overflow boundcpy WrongSizeParam\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=58
Status	New

The size of the buffer used by luaO_chunkid in l, at line 252 of emscripten/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to l, at line 252 of emscripten/lobject.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	256	256

Object	l	l

Code Snippet
File Name emscripten/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
256.        memcpy(out, source + 1, l * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=59
Status	New

The size of the buffer used by luaO_chunkid in char, at line 252 of emscripten/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to char, at line 252 of emscripten/lobject.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	256	256
Object	char	char

Code Snippet
File Name emscripten/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
256.        memcpy(out, source + 1, l * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=60
Status	New

The size of the buffer used by luaO_chunkid in l, at line 252 of emscripten/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to l, at line 252 of emscripten/lobject.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	264	264
Object	l	l

Code Snippet
File Name emscripten/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
264.        memcpy(out, source + 1, l * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=61
Status	New

The size of the buffer used by luaO_chunkid in char, at line 252 of emscripten/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to char, at line 252 of emscripten/lobject.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	264	264
Object	char	char

Code Snippet
File Name emscripten/lobject.c
Method void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
264.        memcpy(out, source + 1, l * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 12:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=62
Status	New

The size of the buffer used by luaO_chunkid in bufflen, at line 252 of emscripten/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to bufflen, at line 252 of emscripten/lobject.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	268	268
Object	bufflen	bufflen

Code Snippet
File Name emscripten/lobject.c

Method	void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
268.        memcpy(out, source + 1 + l - bufflen, bufflen *
sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 13:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=63
Status	New

The size of the buffer used by luaO_chunkid in char, at line 252 of emscripten/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to char, at line 252 of emscripten/lobject.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	268	268
Object	char	char

Code Snippet	
File Name	emscripten/lobject.c
Method	void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
268.        memcpy(out, source + 1 + l - bufflen, bufflen *
sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 14:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=64
Status	New

The size of the buffer used by luaO_chunkid in char, at line 252 of emscripten/lobject.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaO_chunkid passes to char, at line 252 of emscripten/lobject.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	284	284
Object	char	char

Code Snippet	
File Name	emscripten/lobject.c
Method	void luaO_chunkid (char *out, const char *source, size_t bufflen) {

```
....
284.      memcpy(out, POS, (LL(POS) + 1) * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 15:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=65
Status	New

The size of the buffer used by luaV_concat in l, at line 293 of emscripten/lvm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaV_concat passes to l, at line 293 of emscripten/lvm.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/lvm.c	emscripten/lvm.c
Line	324	324
Object	l	l

Code Snippet
File Name emscripten/lvm.c
Method void luaV_concat (lua_State *L, int total) {

```
....
324.          memcpy(buffer+tl, svalue(top-i), l * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 16:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=66
Status	New

The size of the buffer used by luaV_concat in char, at line 293 of emscripten/lvm.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that luaV_concat passes to char, at line 293 of emscripten/lvm.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/lvm.c	emscripten/lvm.c
Line	324	324
Object	char	char

Code Snippet
File Name emscripten/lvm.c
Method void luaV_concat (lua_State *L, int total) {

```
....
324.          memcpy(buffer+tl, svalue(top-i), l * sizeof(char));
```

Buffer Overflow boundcpy WrongSizeParam\Path 17:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=67
Status	New

The size of the buffer used by *ConvexHDup in src, at line 1666 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *ConvexHDup passes to src, at line 1666 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1668	1668
Object	src	src

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHDup(ConvexH *src) {

```
....
1668.       memcpy(dst->vertices.element,src-
>vertices.element,sizeof(float3)*src->vertices.count);
```

Buffer Overflow boundcpy WrongSizeParam\Path 18:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=68
Status	New

The size of the buffer used by *ConvexHDup in float3, at line 1666 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *ConvexHDup passes to float3, at line 1666 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1668	1668
Object	float3	float3

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHDup(ConvexH *src) {

```
....
1668.       memcpy(dst->vertices.element,src-
>vertices.element,sizeof(float3)*src->vertices.count);
```

Buffer Overflow boundcpy WrongSizeParam\Path 19:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=69
Status	New

The size of the buffer used by *ConvexHDup in src, at line 1666 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *ConvexHDup passes to src, at line 1666 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1669	1669
Object	src	src

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHDup(ConvexH *src) {

```
....
1669.       memcpy(dst->edges.element,src-
>edges.element,sizeof(HalfEdge)*src->edges.count);
```

Buffer Overflow boundcpy WrongSizeParam\Path 20:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=70
Status	New

The size of the buffer used by *ConvexHDup in HalfEdge, at line 1666 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *ConvexHDup passes to HalfEdge, at line 1666 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1669	1669
Object	HalfEdge	HalfEdge

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHDup(ConvexH *src) {

```
....
1669.       memcpy(dst->edges.element,src-
>edges.element,sizeof(HalfEdge)*src->edges.count);
```

Buffer Overflow boundcpy WrongSizeParam\Path 21:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=71
Status	New

The size of the buffer used by *ConvexHDup in src, at line 1666 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *ConvexHDup passes to src, at line 1666 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1670	1670
Object	src	src

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHDup(ConvexH *src) {

```
....
1670.       memcpy(dst->facets.element,src-
>facets.element,sizeof(Plane)*src->facets.count);
```

Buffer Overflow boundcpy WrongSizeParam\Path 22:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=72
Status	New

The size of the buffer used by *ConvexHDup in Plane, at line 1666 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *ConvexHDup passes to Plane, at line 1666 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1670	1670
Object	Plane	Plane

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHDup(ConvexH *src) {

```
....
1670.       memcpy(dst->facets.element,src-
>facets.element,sizeof(Plane)*src->facets.count);
```

Buffer Overflow boundcpy WrongSizeParam\Path 23:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=73
Status	New

The size of the buffer used by *ConvexHCrop in under_edge_count, at line 1851 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *ConvexHCrop passes to under_edge_count, at line 1851 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2146	2146
Object	under_edge_count	under_edge_count

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2146.
      memcpy(under.edges.element,tmpunderedges,sizeof(HalfEdge)*under_ed
ge_count);
```

Buffer Overflow boundcpy WrongSizeParam\Path 24:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=74
Status	New

The size of the buffer used by *ConvexHCrop in HalfEdge, at line 1851 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *ConvexHCrop passes to HalfEdge, at line 1851 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2146	2146
Object	HalfEdge	HalfEdge

Code Snippet

File Name	emscripten/cd_hull.cpp
Method	ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2146.
      memcpy(under.edges.element,tmpunderedges,sizeof(HalfEdge)*under_ed
ge_count);
```

Buffer Overflow boundcpy WrongSizeParam\Path 25:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=75
Status	New

The size of the buffer used by *ConvexHCrop in underplanescount, at line 1851 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *ConvexHCrop passes to underplanescount, at line 1851 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2147	2147
Object	underplanescount	underplanescount

Code Snippet

File Name	emscripten/cd_hull.cpp
Method	ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2147.
      memcpy(under.facets.element,tmpunderplanes,sizeof(Plane)*underplan
escount);
```

Buffer Overflow boundcpy WrongSizeParam\Path 26:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=76
Status	New

The size of the buffer used by *ConvexHCrop in Plane, at line 1851 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *ConvexHCrop passes to Plane, at line 1851 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2147	2147
Object	Plane	Plane

Code Snippet	
File Name	emscripten/cd_hull.cpp
Method	ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2147.
      memcpy(under.facets.element,tmpunderplanes,sizeof(Plane)*underplan
escount);
```

Buffer Overflow boundcpy WrongSizeParam\Path 27:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=77
Status	New

The size of the buffer used by HullLibrary::CreateConvexHull in hr, at line 2792 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that HullLibrary::CreateConvexHull passes to hr, at line 2792 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2873	2873
Object	hr	hr

Code Snippet	
File Name	emscripten/cd_hull.cpp
Method	HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2873.                                memcpy(result.mIndices, hr.mIndices,
sizeof(unsigned int)*hr.mIndexCount);
```

Buffer Overflow boundcpy WrongSizeParam\Path 28:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=78
Status	New

The size of the buffer used by HullLibrary::CreateConvexHull in unsigned, at line 2792 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that HullLibrary::CreateConvexHull passes to unsigned, at line 2792 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp

Line	2873	2873
Object	unsigned	unsigned

Code Snippet
File Name emscripten/cd_hull.cpp
Method HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2873.                                memcpy(result.mIndices, hr.mIndices,
sizeof(unsigned int)*hr.mIndexCount);
```

Buffer Overflow boundcpy WrongSizeParam\Path 29:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=79
Status New

The size of the buffer used by HullLibrary::CreateConvexHull in ovcount, at line 2792 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that HullLibrary::CreateConvexHull passes to ovcount, at line 2792 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2853	2853
Object	ovcount	ovcount

Code Snippet
File Name emscripten/cd_hull.cpp
Method HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2853.                           memcpy(result.mOutputVertices, vscratch,
sizeof(float)*3*ovcount );
```

Buffer Overflow boundcpy WrongSizeParam\Path 30:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=80
Status New

The size of the buffer used by HullLibrary::CreateConvexHull in float, at line 2792 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that HullLibrary::CreateConvexHull passes to float, at line 2792 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2853	2853
Object	float	float

Code Snippet
File Name emscripten/cd_hull.cpp
Method HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2853.                          memcpy(result.mOutputVertices, vscratch,
sizeof(float)*3*ovcount );
```

Buffer Overflow boundcpy WrongSizeParam\Path 31:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=81
Status New

The size of the buffer used by HullLibrary::CreateConvexHull in ovcount, at line 2792 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that HullLibrary::CreateConvexHull passes to ovcount, at line 2792 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2884	2884
Object	ovcount	ovcount

Code Snippet
File Name emscripten/cd_hull.cpp
Method HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2884.                          memcpy(result.mOutputVertices, vscratch,
sizeof(float)*3*ovcount );
```

Buffer Overflow boundcpy WrongSizeParam\Path 32:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=82
Status New

The size of the buffer used by HullLibrary::CreateConvexHull in float, at line 2792 of emscripten/cd_hull.cpp, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that HullLibrary::CreateConvexHull passes to float, at line 2792 of emscripten/cd_hull.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2884	2884
Object	float	float

Code Snippet
File Name emscripten/cd_hull.cpp
Method HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2884.                        memcpy(result.mOutputVertices, vscratch,
sizeof(float)*3*ovcount );
```

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=1050074&projectid=50064&pathid=188
Status New

	Source	Destination
File	emscripten/btMultiSapBroadphase.cpp	emscripten/btMultiSapBroadphase.cpp
Line	131	131
Object	bridgeProxyRef	bridgeProxyRef

Code Snippet
File Name emscripten/btMultiSapBroadphase.cpp
Method void btMultiSapBroadphase::addToChildBroadphase(btMultiSapProxy* parentMultiSapProxy, btBroadphaseProxy* childProxy, btBroadphaseInterface* childBroadphase)

```
....
131.        btBridgeProxy* bridgeProxyRef = new(mem) btBridgeProxy;
```

Memory Leak\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=189
Status	New

	Source	Destination
File	emscripten/btMultiSapBroadphase.cpp	emscripten/btMultiSapBroadphase.cpp
Line	50	50
Object	m_overlappingPairs	m_overlappingPairs

Code Snippet
File Name emscripten/btMultiSapBroadphase.cpp
Method btMultiSapBroadphase::btMultiSapBroadphase(int /*maxProxies*/,btOverlappingPairCache* pairCache)

```
....
50.          m_overlappingPairs = new (mem)btSortedOverlappingPairCache();
```

Memory Leak\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=190
Status	New

	Source	Destination
File	emscripten/btMultiSapBroadphase.cpp	emscripten/btMultiSapBroadphase.cpp
Line	71	71
Object	m_filterCallback	m_filterCallback

Code Snippet
File Name emscripten/btMultiSapBroadphase.cpp
Method btMultiSapBroadphase::btMultiSapBroadphase(int /*maxProxies*/,btOverlappingPairCache* pairCache)

```
....
71.    m_filterCallback = new (mem)btMultiSapOverlapFilterCallback();
```

Memory Leak\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=191
Status	New

	Source	Destination
File	emscripten/btMultiSapBroadphase.cpp	emscripten/btMultiSapBroadphase.cpp
Line	90	90
Object	m_optimizedAabbTree	m_optimizedAabbTree

Code Snippet
File Name emscripten/btMultiSapBroadphase.cpp
Method void btMultiSapBroadphase::buildTree(const btVector3& bvhAabbMin,const btVector3& bvhAabbMax)

```
....
90.    m_optimizedAabbTree = new btQuantizedBvh();
```

Memory Leak\Path 5:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=192
Status New

	Source	Destination
File	emscripten/cubegeom_pre.c	emscripten/cubegeom_pre.c
Line	219	219
Object	infoLog	infoLog

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

```
....
219.            char* infoLog = (char *)malloc(sizeof(char) * infoLen+1);
```

Memory Leak\Path 6:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=193
Status New

	Source	Destination
File	emscripten/cubegeom_pre.c	emscripten/cubegeom_pre.c
Line	235	235
Object	infoLog	infoLog

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

```
....
235.            char* infoLog = (char *)malloc(sizeof(char) * infoLen+1);
```

Memory Leak\Path 7:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=194
Status New

	Source	Destination
File	emscripten/cubegeom_pre_vao.c	emscripten/cubegeom_pre_vao.c
Line	225	225
Object	infoLog	infoLog

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

```
....
225.            char* infoLog = (char *)malloc(sizeof(char) * infoLen+1);
```

Memory Leak\Path 8:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=195
Status New

	Source	Destination
File	emscripten/cubegeom_pre_vao.c	emscripten/cubegeom_pre_vao.c
Line	241	241
Object	infoLog	infoLog

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

```
....
241.            char* infoLog = (char *)malloc(sizeof(char) * infoLen+1);
```

Memory Leak\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=196
Status	New

	Source	Destination
File	emscripten/gl_matrix_identity.c	emscripten/gl_matrix_identity.c
Line	42	42
Object	data	data

Code Snippet
File Name emscripten/gl_matrix_identity.c
Method void verify() {

```
....
42.    unsigned char *data = (unsigned char*)malloc(width*height*4);
```

Memory Leak\Path 10:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=197
Status	New

	Source	Destination
File	emscripten/linpack2.c	emscripten/linpack2.c
Line	1134	1134
Object	a	a

Code Snippet
File Name emscripten/linpack2.c
Method double *r8mat_gen (int lda, int n)

```
....
1134.    a = ( double * ) malloc ( lda * n * sizeof ( double ) );
```

Memory Leak\Path 11:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=198
Status	New

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2675	2675
Object	faces_out	faces_out

Code Snippet
File Name emscripten/cd_hull.cpp
Method int overhull(Plane *planes,int planes_count,float3 *verts, int verts_count,int maxplanes,

```
....
2675.       faces_out = (int*)malloc(sizeof(int)*(1+c->facets.count+c->edges.count));     // new int[1+c->facets.count+c->edges.count];
```

Memory Leak\Path 12:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=199
Status New

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2847	2847
Object	mOutputVertices	mOutputVertices

Code Snippet
File Name emscripten/cd_hull.cpp
Method HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2847.                           result.mOutputVertices    = (float *)malloc( sizeof(float)*ovcount*3);
```

Memory Leak\Path 13:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=200
Status New

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2851	2851

Object	mIndices	mIndices

Code Snippet
File Name emscripten/cd_hull.cpp
Method HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2851.                           result.mIndices            = (unsigned int
*) malloc( sizeof(unsigned int)*hr.mIndexCount);
```

Memory Leak\Path 14:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=201
Status New

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2880	2880
Object	mOutputVertices	mOutputVertices

Code Snippet
File Name emscripten/cd_hull.cpp
Method HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2880.                           result.mOutputVertices     = (float
*)malloc( sizeof(float)*ovcount*3);
```

Memory Leak\Path 15:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=202
Status New

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2883	2883
Object	mIndices	mIndices

Code Snippet
File Name emscripten/cd_hull.cpp

Method	HullError HullLibrary::CreateConvexHull(const HullDesc &desc, // describes the input request

```
....
2883.                          result.mIndices           = (unsigned int
*) malloc( sizeof(unsigned int)*result.mNumIndices);
```

Memory Leak\Path 16:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=203
Status	New

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	235	235
Object	vertices	vertices

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
235.        *vertices = malloc ( sizeof(GLfloat) * 3 * numVertices );
```

Memory Leak\Path 17:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=204
Status	New

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	245	245
Object	normals	normals

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
245.        *normals = malloc ( sizeof(GLfloat) * 3 * numVertices );
```

Memory Leak\Path 18:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=205
Status	New

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	251	251
Object	texCoords	texCoords

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
251.        *texCoords = malloc ( sizeof(GLfloat) * 2 * numVertices );
```

Memory Leak\Path 19:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=206
Status	New

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	275	275
Object	indices	indices

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
275.        *indices = malloc ( sizeof(GLushort) * numIndices );
```

Memory Leak\Path 20:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=207
Status	New

	Source	Destination

File	emscripten/esShapes.c	emscripten/esShapes.c
Line	66	66
Object	vertices	vertices

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenSphere (int numSlices, float radius, GLfloat **vertices, GLfloat **normals,

```
....
66.         *vertices = malloc ( sizeof(GLfloat) * 3 * numVertices );
```

Memory Leak\Path 21:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=208
Status New

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	69	69
Object	normals	normals

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenSphere (int numSlices, float radius, GLfloat **vertices, GLfloat **normals,

```
....
69.         *normals = malloc ( sizeof(GLfloat) * 3 * numVertices );
```

Memory Leak\Path 22:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=209
Status New

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	72	72
Object	texCoords	texCoords

Code Snippet

File Name	emscripten/esShapes.c
Method	int ESUTIL_API esGenSphere (int numSlices, float radius, GLfloat **vertices, GLfloat **normals,

```
....
72.         *texCoords = malloc ( sizeof(GLfloat) * 2 * numVertices );
```

Memory Leak\Path 23:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=210
Status	New

	Source	Destination
File	emscripten/esShapes.c	emscripten/esShapes.c
Line	75	75
Object	indices	indices

Code Snippet

File Name	emscripten/esShapes.c
Method	int ESUTIL_API esGenSphere (int numSlices, float radius, GLfloat **vertices, GLfloat **normals,

```
....
75.         *indices = malloc ( sizeof(GLushort) * numIndices );
```

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=1050074&projectid=50064&pathid=107
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination

File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1935	1935
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
1935.
      tmpunderedges[edgeflag[edge0.ea].undermap].ea = under_edge_count;
```

Short Overflow\Path 2:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=108
Status New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1974	1974
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
1974.
      tmpunderedges[edgeflag[edge0.ea].undermap].ea = under_edge_count;
```

Short Overflow\Path 3:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=109
Status New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination

File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2056	2056
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2056.
      tmpunderedges[edgeflag[edge0.ea].undermap].ea = under_edge_count;
```

Short Overflow\Path 4:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=110
Status New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2142	2142
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2142.             tmpunderedges[coplanaredges[i].ea].ea =
under_edge_count+i;
```

Short Overflow\Path 5:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=111
Status New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination

File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1928	1928
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
1928.                          edgeflag[e0].undermap = under_edge_count;
```

Short Overflow\Path 6:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=112
Status New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1950	1950
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
1950.                                  edgeflag[e0].undermap =
under_edge_count;
```

Short Overflow\Path 7:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=113
Status New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp

Line	1967	1967
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
1967.                           edgeflag[e0].undermap = under_edge_count;
```

Short Overflow\Path 8:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=114
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2051	2051
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2051.                           edgeflag[e0].undermap = under_edge_count;
```

Short Overflow\Path 9:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=115
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2096	2096

Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2096.                    coplanaredges[coplanaredges_num].ea =
coplanaredge;
```

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=1050074&projectid=50064&pathid=42
Status	New

The application performs an illegal operation in main, in emscripten/whets.cpp. In line 256, the program attempts to divide by TimeUsed, 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 TimeUsed in main of emscripten/whets.cpp, at line 256.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	302	302
Object	TimeUsed	TimeUsed

Code Snippet
File Name emscripten/whets.cpp
Method int main(int argc, char *argv[])

```
....
302.        if (TimeUsed > 0) xtra = (long)((SPDP)(duration * xtra) /
TimeUsed);
```

Divide By Zero\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=43
Status	New

The application performs an illegal operation in main, in emscripten/whets.cpp. In line 256, the program attempts to divide by BinaryExpr, 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 BinaryExpr in main of emscripten/whets.cpp, at line 256.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	327	327
Object	BinaryExpr	BinaryExpr

Code Snippet
File Name emscripten/whets.cpp
Method int main(int argc, char *argv[])

```
....
327.              mwips=(float)(xtra) * (float)(x100) / (10 * TimeUsed);
```

Divide By Zero\Path 3:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=44
Status New

The application performs an illegal operation in PolyHit, in emscripten/cd_hull.cpp. In line 1321, the program attempts to divide by BinaryExpr, 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 BinaryExpr in PolyHit of emscripten/cd_hull.cpp, at line 1321.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1350	1350
Object	BinaryExpr	BinaryExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method int PolyHit(const float3 *vert, const int n, const float3 &v0, const float3 &v1, float3 *impact, float3 *normal)

```
....
1350.       float a = d0/(d0-d1);
```

Divide By Zero\Path 4:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=45
Status New

The application performs an illegal operation in void btConvexPolyhedron::initialize, in emscripten/btConvexPolyhedron.cpp. In line 75, the program attempts to divide by TotalArea, 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 TotalArea in void btConvexPolyhedron::initialize of emscripten/btConvexPolyhedron.cpp, at line 75.

	Source	Destination
File	emscripten/btConvexPolyhedron.cpp	emscripten/btConvexPolyhedron.cpp
Line	162	162
Object	TotalArea	TotalArea

Code Snippet
File Name emscripten/btConvexPolyhedron.cpp
Method void btConvexPolyhedron::initialize()

```
....
162.        m_localCenter /= TotalArea;
```

Divide By Zero\Path 5:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=46
Status New

The application performs an illegal operation in main, in emscripten/linpack2.c. In line 22, the program attempts to divide by b_max, 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 b_max in main of emscripten/linpack2.c, at line 22.

	Source	Destination
File	emscripten/linpack2.c	emscripten/linpack2.c
Line	183	183
Object	b_max	b_max

Code Snippet
File Name emscripten/linpack2.c
Method int main(int argc, char **argv) {

```
....
183.    residn = resid_max / ( double ) N / a_max / b_max / eps;
```

Divide By Zero\Path 6:

Severity Medium
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=47

Status	New

The application performs an illegal operation in main, in emscripten/linpack2.c. In line 22, the program attempts to divide by a_max, 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 a_max in main of emscripten/linpack2.c, at line 22.

	Source	Destination
File	emscripten/linpack2.c	emscripten/linpack2.c
Line	183	183
Object	a_max	a_max

Code Snippet
File Name emscripten/linpack2.c
Method int main(int argc, char **argv) {

```
....
183.    residn = resid_max / ( double ) N / a_max / b_max / eps;
```

MemoryFree on StackVariable

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

Description

MemoryFree on StackVariable\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=83
Status	New

Calling free() (line 40) on a variable that was not dynamically allocated (line 40) in file emscripten/hsearch.c may result with a crash.

	Source	Destination
File	emscripten/hsearch.c	emscripten/hsearch.c
Line	68	68
Object	oldtab	oldtab

Code Snippet
File Name emscripten/hsearch.c
Method static int resize(size_t nel, struct hsearch_data *htab)

```
....
68.    free(oldtab);
```

MemoryFree on StackVariable\Path 2:

Severity	Medium
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=84
Status	New

Calling free() (line 22) on a variable that was not dynamically allocated (line 22) in file emscripten/linpack2.c may result with a crash.

	Source	Destination
File	emscripten/linpack2.c	emscripten/linpack2.c
Line	140	140
Object	a	a

Code Snippet
File Name emscripten/linpack2.c
Method int main(int argc, char **argv) {

```
....
140.    free ( a );
```

MemoryFree on StackVariable\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=85
Status	New

Calling free() (line 22) on a variable that was not dynamically allocated (line 22) in file emscripten/linpack2.c may result with a crash.

	Source	Destination
File	emscripten/linpack2.c	emscripten/linpack2.c
Line	210	210
Object	a	a

Code Snippet
File Name emscripten/linpack2.c
Method int main(int argc, char **argv) {

```
....
210.    free ( a );
```

MemoryFree on StackVariable\Path 4:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=86

Status	New

Calling free() (line 61) on a variable that was not dynamically allocated (line 61) in file emscripten/ParticleSystem.c may result with a crash.

	Source	Destination
File	emscripten/ParticleSystem.c	emscripten/ParticleSystem.c
Line	83	83
Object	buffer	buffer

Code Snippet
File Name emscripten/ParticleSystem.c
Method GLuint LoadTexture (char *fileName)

```
....
83.     free ( buffer );
```

MemoryFree on StackVariable\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=87
Status	New

Calling free() (line 60) on a variable that was not dynamically allocated (line 60) in file emscripten/ParticleSystem_orig.c may result with a crash.

	Source	Destination
File	emscripten/ParticleSystem_orig.c	emscripten/ParticleSystem_orig.c
Line	82	82
Object	buffer	buffer

Code Snippet
File Name emscripten/ParticleSystem_orig.c
Method GLuint LoadTexture (char *fileName)

```
....
82.     free ( buffer );
```

MemoryFree on StackVariable\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=88
Status	New

Calling free() (line 1469) on a variable that was not dynamically allocated (line 1469) in file emscripten/cd_hull.cpp may result with a crash.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1483	1483
Object	old	old

Code Snippet
File Name emscripten/cd_hull.cpp
Method template <class Type> void Array<Type>::allocate(int s)

```
....
1483.             free(old);
```

Char Overflow

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

Categories

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

Description

Char Overflow\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=95
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	1988	1988
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
1988.                         tmpunderedges[under_edge_count].v = vout;
```

Char Overflow\Path 2:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=96
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2015	2015
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2015.                                   tmpunderedges[under_edge_count].v =
vout;
```

Char Overflow\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=97
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2049	2049
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2049.                           tmpunderedges[under_edge_count].v = vin;
```

Char Overflow\Path 4:

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

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=98
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2085	2085
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2085.                    planeflag[currentplane].undermap =
underplanescount;
```

Char Overflow\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=99
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2097	2097
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2097.                    coplanaredges[coplanaredges_num].v0 = vin;
```

Char Overflow\Path 6:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=100

Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 1851 of emscripten/cd_hull.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2098	2098
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2098.                    coplanaredges[coplanaredges_num].v1 = vout;
```

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=1050074&projectid=50064&pathid=101
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 11 of emscripten/__secs_to_tm.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/__secs_to_tm.c	emscripten/__secs_to_tm.c
Line	32	32
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/__secs_to_tm.c
Method int __secs_to_tm(long long t, struct tm *tm)

```
....
32.    wday = (3+days)%7;
```

Integer Overflow\Path 2:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=102
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 11 of emscripten/__secs_to_tm.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/__secs_to_tm.c	emscripten/__secs_to_tm.c
Line	35	35
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/__secs_to_tm.c
Method int __secs_to_tm(long long t, struct tm *tm)

```
....
35.   qc_cycles = days / DAYS_PER_400Y;
```

Integer Overflow\Path 3:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=103
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 11 of emscripten/__secs_to_tm.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/__secs_to_tm.c	emscripten/__secs_to_tm.c
Line	36	36
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/__secs_to_tm.c
Method int __secs_to_tm(long long t, struct tm *tm)

```
....
36.   remdays = days % DAYS_PER_400Y;
```

Integer Overflow\Path 4:

Severity	Medium

Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=104
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 11 of emscripten/__secs_to_tm.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/__secs_to_tm.c	emscripten/__secs_to_tm.c
Line	26	26
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/__secs_to_tm.c
Method int __secs_to_tm(long long t, struct tm *tm)

```
....
26.   remsecs = secs % 86400;
```

Integer Overflow\Path 5:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=105
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 187 of emscripten/crypt_sha256.c. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/crypt_sha256.c	emscripten/crypt_sha256.c
Line	235	235
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/crypt_sha256.c
Method static char *sha256crypt(const char *key, const char *setting, char *output)

```
....
235.                     r = u;
```

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=1050074&projectid=50064&pathid=50
Status	New

The pointer result at emscripten/psmodule.c in line 397 is being used after it has been freed.

	Source	Destination
File	emscripten/psmodule.c	emscripten/psmodule.c
Line	436	436
Object	result	result

Code Snippet
File Name emscripten/psmodule.c
Method ps_unicodes_char_index(PS_Unicodes table,

```
....
436.         return result->glyph_index;
```

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=1050074&projectid=50064&pathid=89
Status	New

The function newsize in emscripten/hsearch.c at line 40 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	emscripten/hsearch.c	emscripten/hsearch.c
Line	51	51
Object	newsize	newsize

Code Snippet
File Name emscripten/hsearch.c
Method static int resize(size_t nel, struct hsearch_data *htab)

```
....
51.   htab->__tab->entries = calloc(newsize, sizeof *htab->__tab-
>entries);
```

Long Overflow

Query Path:
CPP\Cx\CPP Integer Overflow\Long 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
Long Overflow\Path 1:

Severity	Medium
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=106
Status	New

A variable of a larger data type, AssignExpr, is being assigned to a smaller data type, in 256 of emscripten/whets.cpp. This will cause a loss of data, often the significant bits of a numerical value or the sign bit.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	302	302
Object	AssignExpr	AssignExpr

Code Snippet
File Name emscripten/whets.cpp
Method int main(int argc, char *argv[])

```
....
302.        if (TimeUsed > 0) xtra = (long)((SPDP)(duration * xtra) /
TimeUsed);
```

Use After Free

Query Path:
CPP\Cx\CPP Medium Threat\Use After Free Version:1

Categories

NIST SP 800-53: SC-5 Denial of Service Protection (P1)
OWASP Top 10 2017: A1-Injection

Description
Use After Free\Path 1:

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

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=211
Status	New

The pointer htab at emscripten/hsearch.c in line 119 is being used after it has been freed.

	Source	Destination
File	emscripten/hsearch.c	emscripten/hsearch.c
Line	121	122
Object	entries	htab

Code Snippet
File Name emscripten/hsearch.c
Method static void __hdestroy_r(struct hsearch_data *htab)

```
....
121.        if (htab->__tab) free(htab->__tab->entries);
122.        free(htab->__tab);
```

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=1050074&projectid=50064&pathid=15
Status	New

The *sha256crypt method calls the sprintf function, at line 187 of emscripten/crypt_sha256.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	emscripten/crypt_sha256.c	emscripten/crypt_sha256.c
Line	237	237
Object	sprintf	sprintf

Code Snippet
File Name emscripten/crypt_sha256.c
Method static char *sha256crypt(const char *key, const char *setting, char *output)

```
....
237.              sprintf(rounds, "rounds=%u$", r);
```

Unchecked Return Value\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=16
Status	New

The main method calls the sprintf function, at line 256 of emscripten/whets.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	emscripten/whets.cpp	emscripten/whets.cpp
Line	273	273
Object	sprintf	sprintf

Code Snippet
File Name emscripten/whets.cpp
Method int main(int argc, char *argv[])

```
....
273.              sprintf(timeday, "%s", asctime(localtime(&t)));
```

Unchecked Return Value\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=17
Status	New

The *sha256crypt method calls the p function, at line 187 of emscripten/crypt_sha256.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	emscripten/crypt_sha256.c	emscripten/crypt_sha256.c
Line	297	297
Object	p	p

Code Snippet
File Name emscripten/crypt_sha256.c
Method static char *sha256crypt(const char *key, const char *setting, char *output)

```
....
297.        p += sprintf(p, "$5$%s%.*s$", rounds, slen, salt);
```

Unchecked Return Value\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=18
Status	New

The esGenCube method calls the Pointer function, at line 141 of emscripten/esShapes.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	emscripten/esShapes.c	emscripten/esShapes.c
Line	235	235
Object	Pointer	Pointer

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
235.        *vertices = malloc ( sizeof(GLfloat) * 3 * numVertices );
```

Unchecked Return Value\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=19
Status	New

The esGenCube method calls the Pointer function, at line 141 of emscripten/esShapes.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	emscripten/esShapes.c	emscripten/esShapes.c
Line	245	245
Object	Pointer	Pointer

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
245.        *normals = malloc ( sizeof(GLfloat) * 3 * numVertices );
```

Unchecked Return Value\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=20
Status	New

The esGenCube method calls the Pointer function, at line 141 of emscripten/esShapes.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	emscripten/esShapes.c	emscripten/esShapes.c
Line	251	251
Object	Pointer	Pointer

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
251.        *texCoords = malloc ( sizeof(GLfloat) * 2 * numVertices );
```

Unchecked Return Value\Path 7:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=21
Status	New

The esGenCube method calls the Pointer function, at line 141 of emscripten/esShapes.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	emscripten/esShapes.c	emscripten/esShapes.c
Line	275	275
Object	Pointer	Pointer

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenCube (float scale, GLfloat **vertices, GLfloat **normals,

```
....
275.        *indices = malloc ( sizeof(GLushort) * numIndices );
```

Unchecked Return Value\Path 8:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=22
Status	New

The esGenSphere method calls the Pointer function, at line 54 of emscripten/esShapes.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	emscripten/esShapes.c	emscripten/esShapes.c
Line	66	66
Object	Pointer	Pointer

Code Snippet
File Name emscripten/esShapes.c
Method int ESUTIL_API esGenSphere (int numSlices, float radius, GLfloat **vertices, GLfloat **normals,

```
....
66.        *vertices = malloc ( sizeof(GLfloat) * 3 * numVertices );
```

Unchecked Return Value\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=23
Status	New

The esGenSphere method calls the Pointer function, at line 54 of emscripten/esShapes.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	emscripten/esShapes.c	emscripten/esShapes.c
Line	69	69
Object	Pointer	Pointer

Code Snippet
File Name emscripten/esShapes.c

Method	int ESUTIL_API esGenSphere (int numSlices, float radius, GLfloat **vertices, GLfloat **normals,

```
....
69.         *normals = malloc ( sizeof(GLfloat) * 3 * numVertices );
```

Unchecked Return Value\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=24
Status	New

The esGenSphere method calls the Pointer function, at line 54 of emscripten/esShapes.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	emscripten/esShapes.c	emscripten/esShapes.c
Line	72	72
Object	Pointer	Pointer

Code Snippet	
File Name	emscripten/esShapes.c
Method	int ESUTIL_API esGenSphere (int numSlices, float radius, GLfloat **vertices, GLfloat **normals,

```
....
72.         *texCoords = malloc ( sizeof(GLfloat) * 2 * numVertices );
```

Unchecked Return Value\Path 11:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=25
Status	New

The esGenSphere method calls the Pointer function, at line 54 of emscripten/esShapes.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	emscripten/esShapes.c	emscripten/esShapes.c
Line	75	75
Object	Pointer	Pointer

Code Snippet

File Name	emscripten/esShapes.c
Method	int ESUTIL_API esGenSphere (int numSlices, float radius, GLfloat **vertices, GLfloat **normals,

```
....
75.         *indices = malloc ( sizeof(GLushort) * numIndices );
```

Unchecked Return Value\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=26
Status	New

The main method calls the b function, at line 22 of emscripten/linpack2.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	emscripten/linpack2.c	emscripten/linpack2.c
Line	86	86
Object	b	b

Code Snippet	
File Name	emscripten/linpack2.c
Method	int main(int argc, char **argv) {

```
....
86.     b = ( double * ) malloc ( N * sizeof ( double ) );
```

Unchecked Return Value\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=27
Status	New

The main method calls the ipvt function, at line 22 of emscripten/linpack2.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	emscripten/linpack2.c	emscripten/linpack2.c
Line	87	87
Object	ipvt	ipvt

Code Snippet	
File Name	emscripten/linpack2.c
Method	int main(int argc, char **argv) {

```
....
87.     ipvt = ( int * ) malloc ( N * sizeof ( int ) );
```

Unchecked Return Value\Path 14:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=28
Status	New

The main method calls the resid function, at line 22 of emscripten/linpack2.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	emscripten/linpack2.c	emscripten/linpack2.c
Line	88	88
Object	resid	resid

Code Snippet
File Name emscripten/linpack2.c
Method int main(int argc, char **argv) {

```
....
88.     resid = ( double * ) malloc ( N * sizeof ( double ) );
```

Unchecked Return Value\Path 15:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=29
Status	New

The main method calls the rhs function, at line 22 of emscripten/linpack2.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	emscripten/linpack2.c	emscripten/linpack2.c
Line	89	89
Object	rhs	rhs

Code Snippet
File Name emscripten/linpack2.c
Method int main(int argc, char **argv) {

```
....
89.     rhs = ( double * ) malloc ( N * sizeof ( double ) );
```

Unchecked Return Value\Path 16:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=30
Status	New

The main method calls the x function, at line 22 of emscripten/linpack2.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	emscripten/linpack2.c	emscripten/linpack2.c
Line	90	90
Object	x	x

Code Snippet
File Name emscripten/linpack2.c
Method int main(int argc, char **argv) {

```
....
90.     x = ( double * ) malloc ( N * sizeof ( double ) );
```

Unchecked Return Value\Path 17:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=31
Status	New

The *r8mat_gen method calls the a function, at line 1108 of emscripten/linpack2.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	emscripten/linpack2.c	emscripten/linpack2.c
Line	1134	1134
Object	a	a

Code Snippet
File Name emscripten/linpack2.c
Method double *r8mat_gen (int lda, int n)

```
....
1134.    a = ( double * ) malloc ( lda * n * sizeof ( double ) );
```

Unchecked Return Value\Path 18:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=32
Status	New

The main method calls the pLook function, at line 1139 of emscripten/speedtest1.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	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	1311	1311
Object	pLook	pLook

Code Snippet
File Name emscripten/speedtest1.c
Method int main(int argc, char **argv){

```
....
1311.      pLook = malloc( nLook*szLook );
```

Unchecked Return Value\Path 19:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=33
Status	New

The overhull method calls the faces_out function, at line 2636 of emscripten/cd_hull.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	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2675	2675
Object	faces_out	faces_out

Code Snippet
File Name emscripten/cd_hull.cpp
Method int overhull(Plane *planes,int planes_count,float3 *verts, int verts_count,int maxplanes,

```
....
2675.       faces_out = (int*)malloc(sizeof(int)*(1+c->facets.count+c-
>edges.count));     // new int[1+c->facets.count+c->edges.count];
```

Unchecked Return Value\Path 20:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=34
Status	New

The main method calls the infoLog function, at line 37 of emscripten/cubegeom_pre.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	emscripten/cubegeom_pre.c	emscripten/cubegeom_pre.c
Line	219	219
Object	infoLog	infoLog

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

```
....
219.           char* infoLog = (char *)malloc(sizeof(char) * infoLen+1);
```

Unchecked Return Value\Path 21:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=35
Status	New

The main method calls the infoLog function, at line 37 of emscripten/cubegeom_pre.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	emscripten/cubegeom_pre.c	emscripten/cubegeom_pre.c
Line	235	235
Object	infoLog	infoLog

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

```
....
235.            char* infoLog = (char *)malloc(sizeof(char) * infoLen+1);
```

Unchecked Return Value\Path 22:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=36
Status	New

The main method calls the infoLog function, at line 37 of emscripten/cubegeom_pre_vao.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	emscripten/cubegeom_pre_vao.c	emscripten/cubegeom_pre_vao.c
Line	225	225
Object	infoLog	infoLog

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

```
....
225.            char* infoLog = (char *)malloc(sizeof(char) * infoLen+1);
```

Unchecked Return Value\Path 23:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=37
Status	New

The main method calls the infoLog function, at line 37 of emscripten/cubegeom_pre_vao.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	emscripten/cubegeom_pre_vao.c	emscripten/cubegeom_pre_vao.c
Line	241	241
Object	infoLog	infoLog

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

```
....
241.            char* infoLog = (char *)malloc(sizeof(char) * infoLen+1);
```

Unchecked Return Value\Path 24:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=38
Status	New

The verify method calls the data function, at line 40 of emscripten/gl_matrix_identity.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	emscripten/gl_matrix_identity.c	emscripten/gl_matrix_identity.c
Line	42	42
Object	data	data

Code Snippet
File Name emscripten/gl_matrix_identity.c
Method void verify() {

```
....
42.    unsigned char *data = (unsigned char*)malloc(width*height*4);
```

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=1050074&projectid=50064&pathid=119
Status	New

	Source	Destination
File	emscripten/linpack2.c	emscripten/linpack2.c
Line	361	361
Object	iy	iy

Code Snippet
File Name emscripten/linpack2.c
Method void daxpy (int n, double da, double dx[], int incx, double dy[], int incy)

```
....
361.        dy[iy] = dy[iy] + da * dx[ix];
```

Unchecked Array Index\Path 2:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=120
Status New

	Source	Destination
File	emscripten/linpack2.c	emscripten/linpack2.c
Line	813	813
Object	ix	ix

Code Snippet
File Name emscripten/linpack2.c
Method void dscal (int n, double sa, double x[], int incx)

```
....
813.        x[ix] = sa * x[ix];
```

Unchecked Array Index\Path 3:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=121
Status New

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	515	515
Object	len	len

Code Snippet
File Name emscripten/speedtest1.c
Method void testset_main(void){

```
....
515.      zNum[len] = '%';
```

Unchecked Array Index\Path 4:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=122
Status	New

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	535	535
Object	len	len

Code Snippet
File Name emscripten/speedtest1.c
Method void testset_main(void){

```
....
535.        zNum[len] = '%';
```

Unchecked Array Index\Path 5:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=123
Status	New

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	554	554
Object	len	len

Code Snippet
File Name emscripten/speedtest1.c
Method void testset_main(void){

```
....
554.        zNum[len] = '%';
```

Unchecked Array Index\Path 6:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=124
Status	New

	Source	Destination

File	emscripten/t1parse.c	emscripten/t1parse.c
Line	460	460
Object	len	len

Code Snippet
File Name emscripten/t1parse.c
Method T1_Get_Private_Dict(T1_Parser parser,

```
....
460.          parser->private_dict[len] = '\0';
```

Unchecked Array Index\Path 7:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=125
Status New

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	581	581
Object	section	section

Code Snippet
File Name emscripten/whets.cpp
Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
581.        loop_time[section] = time;
```

Unchecked Array Index\Path 8:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=126
Status New

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	587	587
Object	section	section

Code Snippet
File Name emscripten/whets.cpp
Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
587.              results[section] = checknum;
```

Unchecked Array Index\Path 9:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=127
Status	New

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	603	603
Object	section	section

Code Snippet
File Name emscripten/whets.cpp
Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
603.                    loop_mops[section] = 99999;
```

Unchecked Array Index\Path 10:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=128
Status	New

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	604	604
Object	section	section

Code Snippet
File Name emscripten/whets.cpp
Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
604.                    loop_mflops[section] = mflops;
```

Unchecked Array Index\Path 11:

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

	BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=129
Status	New

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	618	618
Object	section	section

Code Snippet
File Name emscripten/whets.cpp
Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
618.                    loop_mops[section] = mops;
```

Unchecked Array Index\Path 12:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=130
Status	New

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	619	619
Object	section	section

Code Snippet
File Name emscripten/whets.cpp
Method void pout(const char title[18], float ops, int type, SPDP checknum,

```
....
619.                    loop_mflops[section] = 0;
```

Unchecked Array Index\Path 13:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=131
Status	New

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	558	558

Object	j	j

Code Snippet
File Name emscripten/whets.cpp
Method void po(SPDP e1[4], long j, long k, long l)

```
....
558.        e1[j] = e1[k];
```

Unchecked Array Index\Path 14:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=132
Status New

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	559	559
Object	k	k

Code Snippet
File Name emscripten/whets.cpp
Method void po(SPDP e1[4], long j, long k, long l)

```
....
559.        e1[k] = e1[l];
```

Unchecked Array Index\Path 15:

Severity Low
Result State To Verify
Online Results http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=133
Status New

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	560	560
Object	l	l

Code Snippet
File Name emscripten/whets.cpp
Method void po(SPDP e1[4], long j, long k, long l)

```
....
560.        e1[l] = e1[j];
```

Unchecked Array Index\Path 16:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=134
Status	New

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2086	2086
Object	underplanescount	underplanescount

Code Snippet
File Name emscripten/cd_hull.cpp
Method ConvexH *ConvexHCrop(ConvexH &convex,const Plane &slice)

```
....
2086.                    tmpunderplanes[underplanescount] =
convex.facets[currentplane];
```

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=1050074&projectid=50064&pathid=39
Status	New

	Source	Destination
File	emscripten/lgc.c	emscripten/lgc.c
Line	471	471
Object	sizeof	sizeof

Code Snippet
File Name emscripten/lgc.c
Method static lu_mem traversetable (global_State *g, Table *h) {

```
....
471.                            sizeof(Proto *) * f->sizep +
```

Use of Sizeof On a Pointer Type\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=40
Status	New

	Source	Destination
File	emscripten/lgc.c	emscripten/lgc.c
Line	1038	1038
Object	sizeof	sizeof

Code Snippet
File Name emscripten/lgc.c
Method static lu_mem singlestep (lua_State *L) {

```
....
1038.         g->GCmemtrav = g->strt.size * sizeof(GCObject*);
```

Use of Sizeof On a Pointer Type\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=41
Status	New

	Source	Destination
File	emscripten/lobject.c	emscripten/lobject.c
Line	208	208
Object	sizeof	sizeof

Code Snippet
File Name emscripten/lobject.c
Method const char *luaO_pushvfstring (lua_State *L, const char *fmt, va_list argp) {

```
....
208.           char buff[4*sizeof(void *) + 8]; /* should be enough space
for a `%p' */
```

Sizeof Pointer Argument

Query Path:
CPP\Cx\CPP Low Visibility\Sizeof Pointer Argument Version:0

Description

Sizeof Pointer Argument\Path 1:

Severity	Low
Result State	To Verify

Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=116
Status	New

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	136	136
Object	aMult	sizeof

Code Snippet
File Name emscripten/speedtest1.c
Method static int integerValue(const char *zArg){

```
....
136.    for(i=0; i<sizeof(aMult)/sizeof(aMult[0]); i++){
```

Sizeof Pointer Argument\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=117
Status	New

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	136	136
Object	aMult	sizeof

Code Snippet
File Name emscripten/speedtest1.c
Method static int integerValue(const char *zArg){

```
....
136.    for(i=0; i<sizeof(aMult)/sizeof(aMult[0]); i++){
```

Sizeof Pointer Argument\Path 3:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=118
Status	New

	Source	Destination
File	emscripten/crypt_blowfish.c	emscripten/crypt_blowfish.c

Line	799	799
Object	ai	sizeof

Code Snippet
File Name emscripten/crypt_blowfish.c
Method char *__crypt_blowfish(const char *key, const char *setting, char *output)

```
....
799.                    !memcmp(ai, yi, sizeof(ai));
```

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=1050074&projectid=50064&pathid=48
Status	New

The buffer allocated by <= in emscripten/__secs_to_tm.c at line 11 does not correctly account for the actual size of the value, resulting in an incorrect allocation that is off by one.

	Source	Destination
File	emscripten/__secs_to_tm.c	emscripten/__secs_to_tm.c
Line	60	60
Object	<=	<=

Code Snippet
File Name emscripten/__secs_to_tm.c
Method int __secs_to_tm(long long t, struct tm *tm)

```
....
60.    for (months=0; days_in_month[months] <= remdays; months++)
```

Potential Off by One Error in Loops\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=49
Status	New

The buffer allocated by <= in emscripten/cd_hull.cpp at line 2231 does not correctly account for the actual size of the value, resulting in an incorrect allocation that is off by one.

	Source	Destination
File	emscripten/cd_hull.cpp	emscripten/cd_hull.cpp
Line	2241	2241
Object	<=	<=

Code Snippet
File Name emscripten/cd_hull.cpp
Method int maxdirsterid(const T *p,int count,const T &dir,Array<int> &allow)

```
....
2241.             for(float x = 0.0f ; x<= 360.0f ; x+= 45.0f)
```

Potential Precision Problem

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

Categories

NIST SP 800-53: SI-10 Information Input Validation (P1)
OWASP Top 10 2017: A1-Injection

Description
Potential Precision Problem\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=92
Status	New

The size of the buffer used by *sha256crypt in "5%s%.*s$", at line 187 of emscripten/crypt_sha256.c, is not properly verified before writing data to the buffer. This can enable a buffer overflow attack, using the source buffer that *sha256crypt passes to "5%s%.*s$", at line 187 of emscripten/crypt_sha256.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/crypt_sha256.c	emscripten/crypt_sha256.c
Line	297	297
Object	"5%s%.*s$"	"5%s%.*s$"

Code Snippet
File Name emscripten/crypt_sha256.c
Method static char *sha256crypt(const char *key, const char *setting, char *output)

```
....
297.         p += sprintf(p, "$5$%s%.*s$", rounds, slen, salt);
```

Potential Precision Problem\Path 2:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=93
Status	New

The size of the buffer used by main in "%s", at line 256 of emscripten/whets.cpp, 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 "%s", at line 256 of emscripten/whets.cpp, to overwrite the target buffer.

	Source	Destination
File	emscripten/whets.cpp	emscripten/whets.cpp
Line	273	273
Object	"%s"	"%s"

Code Snippet
File Name emscripten/whets.cpp
Method int main(int argc, char *argv[])

```
....
273.              sprintf(timeday, "%s", asctime(localtime(&t)));
```

Inconsistent Implementations

Query Path:
CPP\Cx\CPP Low Visibility\Inconsistent Implementations Version:0

Description

Inconsistent Implementations\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=13
Status	New

	Source	Destination
File	emscripten/test_getopt_long.c	emscripten/test_getopt_long.c
Line	30	30
Object	getopt_long	getopt_long

Code Snippet
File Name emscripten/test_getopt_long.c
Method int main(int argc, char **argv) {

```
....
30.      c = getopt_long(argc, argv, "abc:d:012", long_options,
&option_index);
```

Use of Insufficiently Random Values

Query Path:
CPP\Cx\CPP Low Visibility\Use of Insufficiently Random Values Version:0

Categories

FISMA 2014: Media Protection
NIST SP 800-53: SC-28 Protection of Information at Rest (P1)
OWASP Top 10 2017: A3-Sensitive Data Exposure

Description

Use of Insufficiently Random Values\Path 1:

Severity	Low
Result State	To Verify
Online Results	http://WIN-BA8RD5TJ8IG/CxWebClient/ViewerMain.aspx?scanid=1050074&projectid=50064&pathid=14
Status	New

Method Init at line 91 of emscripten/ParticleSystem_orig.c uses a weak method srand to produce random values. These values might be used for secret values, personal identifiers or cryptographic input, allowing an attacker to guess the value.

	Source	Destination
File	emscripten/ParticleSystem_orig.c	emscripten/ParticleSystem_orig.c
Line	149	149
Object	srand	srand

Code Snippet
File Name emscripten/ParticleSystem_orig.c
Method int Init (ESContext *esContext)

```
....
149.     srand ( 0 );
```

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=1050074&projectid=50064&pathid=94
Status	New

The size of the buffer used by main in nHeap, at line 1139 of emscripten/speedtest1.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 1139 of emscripten/speedtest1.c, to overwrite the target buffer.

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	1139	1283
Object	argv	nHeap

Code Snippet
File Name emscripten/speedtest1.c
Method int main(int argc, char **argv){

```
....
1139.  int main(int argc, char **argv){
....
1283.      pHeap = malloc( nHeap );
```

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=1050074&projectid=50064&pathid=212
Status	New

	Source	Destination
File	emscripten/speedtest1.c	emscripten/speedtest1.c
Line	393	393
Object	fprintf	fprintf

Code Snippet
File Name emscripten/speedtest1.c
Method static void traceCallback(void *NotUsed, const char *zSql){

```
....
393.    fprintf(stderr,"%.*s;\n", n, zSql);
```

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 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

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

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

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;
}
```

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);
```

Char Overflow

Risk

What might happen

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

Cause

How does it happen

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

General Recommendations

How to avoid it

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

---

Source Code Examples

CPP

Unsafe Downsize Casting

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

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

    return total;
}
```

Safer Use of Proper Data Types

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

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

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

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

    return total;
}
```

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

Long 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

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 After Free

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
//..
```

Use of Function with Inconsistent Implementations

Weakness ID: 474 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The code uses a function that has inconsistent implementations across operating systems and versions, which might cause security-relevant portability problems.

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

C: *(Often)*

PHP: *(Often)*

All

Potential Mitigations

Do not accept inconsistent behavior from the API specifications when the deviant behavior increase the risk level.

Other Notes

The behavior of functions in this category varies by operating system, and at times, even by operating system version. Implementation differences can include:

- Slight differences in the way parameters are interpreted leading to inconsistent results.
- Some implementations of the function carry significant security risks.
- The function might not be defined on all platforms.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	398	Indicator of Poor Code Quality	**Development Concepts (primary)699 Seven Pernicious Kingdoms (primary)700 Research Concepts (primary)1000**
ParentOf	Weakness Variant	589	Call to Non-ubiquitous API	**Research Concepts (primary)1000**

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Inconsistent Implementations

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	7 Pernicious Kingdoms		Externally Mined
Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-07-01	Eric Dalci	Cigital	External
	updated Potential Mitigations, Time of Introduction		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Relationships, Other Notes, Taxonomy Mappings		
Previous Entry Names			
Change Date	**Previous Entry Name**		
2008-04-11	Inconsistent Implementations		

BACK TO TOP

Use of Insufficiently Random Values

Risk

What might happen

Random values are often used as a mechanism to prevent malicious users from guessing a value, such as a password, encryption key, or session identifier. Depending on what this random value is used for, an attacker would be able to predict the next numbers generated, or previously generated values. This could enable the attacker to hijack another user's session, impersonate another user, or crack an encryption key (depending on what the pseudo-random value was used for).

Cause

How does it happen

The application uses a weak method of generating pseudo-random values, such that other numbers could be determined from a relatively small sample size. Since the pseudo-random number generator used is designed for statistically uniform distribution of values, it is approximately deterministic. Thus, after collecting a few generated values (e.g. by creating a few individual sessions, and collecting the sessionids), it would be possible for an attacker to calculate another sessionid.
Specifically, if this pseudo-random value is used in any security context, such as passwords, keys, or secret identifiers, an attacker would be able to predict the next numbers generated, or previously generated values.

General Recommendations

How to avoid it

Generic Guidance:

- Whenever unpredicatable numbers are required in a security context, use a cryptographically strong random number generator, instead of a statistical pseudo-random generator.
- Use the cryptorandom generator that is built-in to your language or platform, and ensure it is securely seeded. Do not seed the generator with a weak, non-random seed. (In most cases, the default is securely random).
- Ensure you use a long enough random value, to make brute-force attacks unfeasible.

Specific Recommendations:

- Do not use the statistical pseudo-random number generator, use the cryptorandom generator instead. In Java, this is the SecureRandom class.

Source Code Examples

Java

Use of a weak pseudo-random number generator

```
Random random = new Random();

long sessNum = random.nextLong();

String sessionId = sessNum.toString();
```

```
```

Cryptographically secure random number generator

```
SecureRandom random = new SecureRandom();

byte sessBytes[] = new byte[32];

random.nextBytes(sessBytes);

String sessionId = new String(sessBytes);
```

Objc

Use of a weak pseudo-random number generator

```
long sessNum = rand();
NSString* sessionId = [NSString stringWithFormat:@"%ld", sessNum];
```

Cryptographically secure random number generator

```
UInt32 sessBytes;
SecRandomCopyBytes(kSecRandomDefault, sizeof(sessBytes), (uint8_t*)&sessBytes);

NSString* sessionId = [NSString stringWithFormat:@"%llu", sessBytes];
```

Swift

Use of a weak pseudo-random number generator

```
let sessNum = rand();
let sessionId = String(format:"%ld", sessNum)
```

Cryptographically secure random number generator

```
var sessBytes: UInt32 = 0
withUnsafeMutablePointer(&sessBytes, { (sessBytesPointer) -> Void in
    let castedPointer = unsafeBitCast(sessBytesPointer, UnsafeMutablePointer<UInt8>.self)
    SecRandomCopyBytes(kSecRandomDefault, sizeof(UInt32), castedPointer)
})

let sessionId = String(format:"%llu", sessBytes)
```

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
```

Potential Precision Problem

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

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

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

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

Scanned Languages

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