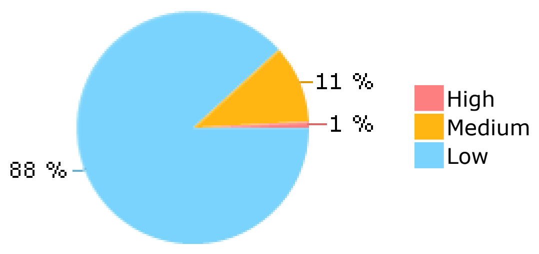
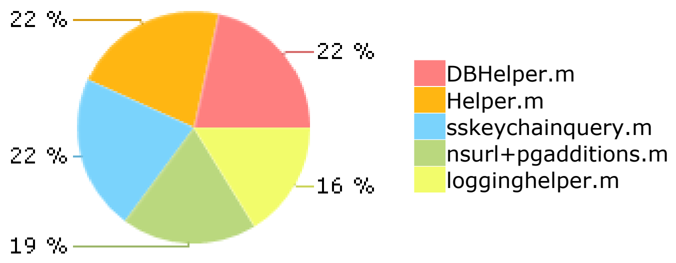
NASA ISS FIT Scan Report

Project Name	NASA ISS FIT
Scan Start	Tuesday, May 12, 2015 3:25:47 PM
Preset	Default 2014
Scan Time	00h:23m:02s
Lines Of Code Scanned	73,488
Files Scanned	310
Report Creation Time	Tuesday, May 12, 2015 3:50:20 PM
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441
Team	appirio
Checkmarx Version	7.1.8 HF1
Scan Type	Full
Source Origin	LocalPath
Density	2/1000 (Vulnerabilities/LOC)
Scan Comments	

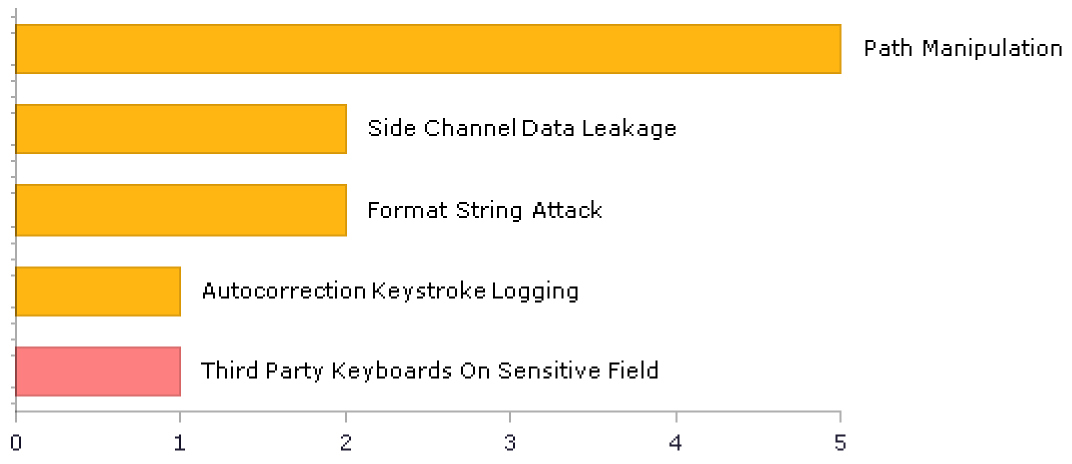
Result Summary



Most Vulnerable Files



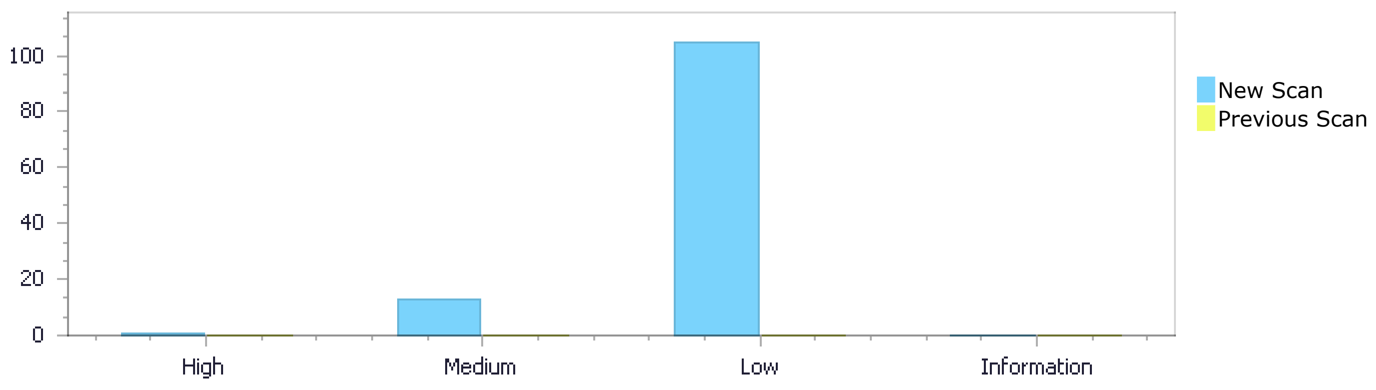
Top 5 Vulnerabilities



Results Distribution By Status

First scan of the project

	High	Medium	Low	Information	Total
New Issues	1	13	105	0	119
Recurrent Issues	0	0	0	0	0
Total	1	13	105	0	119
Fixed Issues	0	0	0	0	0



Results Distribution By State

	High	Medium	Low	Information	Total
To Verify	1	13	105	0	119
Not Exploitable	0	0	0	0	0
Confirmed	0	0	0	0	0
Urgent	0	0	0	0	0
Total	1	13	105	0	119

Result Summary

Vulnerability Type	Occurrences	Severity
Third Party Keyboards On Sensitive Field	1	High
Path Manipulation	5	Medium
Format String Attack	2	Medium
Side Channel Data Leakage	2	Medium
Autocorrection Keystroke Logging	1	Medium
Cut And Paste Leakage	1	Medium
Insecure Data Storage	1	Medium
Screen Caching	1	Medium
Unscrubbed Secret	50	Low
Jailbrake File Referenced By Name	34	Low
Jailbreak Unchecked File Operation Result Code	4	Low
Use of Hardcoded Password	4	Low

Buffer Size Literal Condition	3	Low
Unchecked Return Value	3	Low
Incorrect Initialization	2	Low
Log Forging	2	Low
Functions Apple Recommends To Avoid	1	Low
Unchecked CString Convertion	1	Low
Use of Insufficiently Random Values	1	Low

10 Most Vulnerable Files

High and Medium Vulnerabilities

File Name	Issues Found
/source code/foodintaketracker/controller/loginviewcontroller.h	5
/source code/Helpers/logginghelper.m	4
/source code/foodintaketracker/controller/loginviewcontroller.m	2
/source code/Services/userserviceimpl.m	2
/source code/foodintaketracker/controller/addfoodviewcontroller.m	2
/source code/Services/foodproductserviceimpl.m	2
/source code/Helpers/DBHelper.m	2
/source code/Helpers/datahelper.m	2
/source code/foodintaketracker/controller/consumptionviewcontroller.m	1
/source code/foodintaketracker/common/Helper.m	1

Scanned Queries

Query Name	Issues Found
Unscrubbed Secret	55
Jailbrake File Referenced By Name	34
Path Manipulation	5
Jailbreak Unchecked File Operation Result Code	4
Use of Hardcoded Password	4
Buffer Size Literal Condition	3
Unchecked Return Value	3
Format String Attack	2
Incorrect Initialization	2
Log Forging	2
Side Channel Data Leakage	2
Autocorrection Keystroke Logging	1
Client DOM Open Redirect	1
Client Path Manipulation	1
Client Potential DOM Open Redirect	1
Cut And Paste Leakage	1
DOM XSRF	1
Functions Apple Recommends To Avoid	1
Improper Certificate Validation	1
Insecure Data Storage	1
Insufficient Transport Layer Input	1

Insufficient Transport Layer Output	1
Missing Encryption of Sensitive Data	1
Screen Caching	1
Third Party Keyboards On Sensitive Field	1
Unchecked CString Convertion	1
Use of Insufficiently Random Values	1
Buffer Size Literal Overflow	0
Client Cookies Inspection	0
Client Cross Frame Scripting Attack	0
Client Cross Session Contamination	0
Client DB Parameter Tampering	0
Client DOM Code Injection	0
Client DOM Cookie Poisoning	0
Client DOM Stored Code Injection	0
Client DOM Stored XSS	0
Client DOM XSRF	0
Client DOM XSS	0
Client DoS By Sleep	0
Client Empty Password	0
Client Header Manipulation	0
Client HTML5 Heuristic Session Insecure Storage	0
Client HTML5 Information Exposure	0
Client HTML5 Insecure Storage	0
Client HTML5 Store Sensitive data In Web Storage	0
Client Insecure Randomness	0
Client Insufficient ClickJacking Protection	0
Client Insufficient Key Size	0
Client JQuery Deprecated Symbols	0
Client Located JQuery Outdated Lib File	0
Client Negative Content Length	0
Client Overly Permissive Message Posting	0
Client Password In Comment	0
Client Potential Ad Hoc Ajax	0
Client Potential Code Injection	0
Client Potential ReDoS In Match	0
Client Potential ReDoS In Replace	0
Client Potential XSS	0
Client Privacy Violation	0
Client ReDoS From Regex Injection	0
Client ReDoS In Match	0
Client ReDos In RegExp	0
Client ReDoS In Replace	0
Client Regex Injection	0
Client Remote File Inclusion	0
Client Resource Injection	0
Client Sandbox Allows Scripts With Same Origin	0
Client Second Order Sql Injection	0
Client Server Empty Password	0
Client SQL Injection	0

Client Untrusted Activex	0
Client Use Of Deprecated SQL Database	0
Client Use Of Iframe Without Sandbox	0
Client Use Of JQuery Outdated Version	0
Client Weak Cryptographic Hash	0
Client Weak Encryption	0
Client Weak Password Authentication	0
Client XPATH Injection	0
Code Injection	0
Cookies Inspection	0
Divide By Zero	0
DOM Code Injection	0
DOM Cookie Poisoning	0
DOM Open Redirect	0
DOM XSS	0
Empty Password	0
Hardcoded Absolute Path	0
Hardcoded password in Connection String	0
Heap Inspection	0
HTTP Response Splitting	0
Improper Implementation of NSSecureCoding	0
Improper Resource Shutdown or Release	0
Information Exposure Through an Error Message	0
Information Exposure Through Extension	0
Information Exposure Through Query String	0
Insufficient Encryption Key Size	0
Memory Leak	0
NSPredicate Injection	0
NSPredicate Injection Via Deserialization	0
Null Password	0
Parameter Tampering	0
Path Traversal	0
Plaintext Storage of a Password	0
Poor Authorization and Authentication	0
Potential ReDoS	0
ReDoS	0
Reflected XSS	0
Reflected XSS All Clients	0
Second Order SQL Injection	0
Sensitive Data In Temp Folders	0
Signed Memory Arithmetic	0
SQL Injection	0
SSL Verification Bypass	0
Stored Code Injection	0
Stored Path Traversal	0
Stored XSS	0
Third Party Keyboard Enabled	0
Uncontrolled Format String	0
Unsafe Reflection	0

Unsecure Deserialization	0
URL Injection	0
Use of Broken or Risky Cryptographic Algorithm	0
Use of Deprecated or Obsolete Functions	0
Use of Hardcoded Cryptographic Key	0
Use of Obsolete Functions	0
VF Remoting Client Potential Code Injection	0
VF Remoting Client Potential XSRF	0
VF Remoting Client Potential XSS	0
Weak Password Authentication	0
XML External Entity	0

Scan Results Details

Number of results is limited to 50 for each vulnerability. To get more results please change a setting "Limit results to 50" in report creation wizard.

Third Party Keyboards On Sensitive Field

Description

Third Party Keyboards On Sensitive Field\Path 132:

Severity	High
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=132
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/loginviewcontroller.h	/source code/foodintaketracker/controller/loginviewcontroller.h
Line	92	92
Object	txtPassword	txtPassword

Code Snippet

File Name	/source code/foodintaketracker/controller/loginviewcontroller.h
Method	@property (weak, nonatomic) IBOutlet UITextField *txtPassword;

```
....
92.  @property (weak, nonatomic) IBOutlet UITextField *txtPassword;
```

Path Manipulation

Description

Path Manipulation\Path 67:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=67
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/DBHelper.m	/source code/Helpers/DBHelper.m
Line	177	204
Object	dictionaryWithContentsOfFile:	copyItemAtPath:toPath:error:

Code Snippet

File Name	/source code/Helpers/DBHelper.m
Method	dispatch_once(&onceToken, ^{

```
....
177.          NSDictionary *configuration = [NSDictionary
dictionaryWithContentsOfFile:configBundle];
....
204.                  [[NSFileManager defaultManager]
copyItemAtPath:[localFolder stringByAppendingPathComponent:file]
```

Path Manipulation\Path 68:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=68
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/DBHelper.m	/source code/Helpers/datahelper.m
Line	177	156
Object	dictionaryWithContentsOfFile:	fileExistsAtPath:

Code Snippet

File Name /source code/Helpers/DBHelper.m

Method dispatch_once(&onceToken, ^{

```
....
177.          NSDictionary *configuration = [NSDictionary
dictionaryWithContentsOfFile:configBundle];
```

File Name /source code/Helpers/datahelper.m

Method + (NSString *)getAbsoulteLocalDirectory:(NSString *)localDirectory {

```
....
156.      if(![[NSFileManager defaultManager]
fileExistsAtPath:localDirFullPath]) {
```

Path Manipulation\Path 69:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=69
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/consumptionviewcontroller.m	/source code/foodintaketracker/common/Helper.m
Line	1897	120

Object	dataWithContentsOfFile:options:error:	writeToFile:atomically:

Code Snippet
File Name /source code/foodintaketracker/controller/consumptionviewcontroller.m
Method - (void)audioRecorderEndInterruption:(AVAudioRecorder *)aRecorder withOptions:(NSUInteger)flags {

```
....
1897.       NSData *audioData = [NSData dataWithContentsOfFile:[url path]
options: 0 error:&err];
```

File Name /source code/foodintaketracker/common/Helper.m
Method + (NSString *)saveVoiceRecording:(NSData *)data {

```
....
120.       [data writeToFile:filePath atomically:YES];
```

Path Manipulation\Path 70:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=70
Result Comment	
Status	New

	Source	Destination
File	/source code/issfoodintaketrackertests/basetests.m	/source code/issfoodintaketrackertests/basetests.m
Line	102	102
Object	NSManagedObjectModel	initWithContentsOfURL:

Code Snippet
File Name /source code/issfoodintaketrackertests/basetests.m
Method - (NSManagedObjectModel *)managedObjectModel

```
....
102.       _managedObjectModel = [[NSManagedObjectModel alloc]
initWithContentsOfURL:modelURL];
```

Path Manipulation\Path 71:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=71
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/basetests.m	/source code/foodintaketrackertests/basetests.m
Line	102	102
Object	NSManagedObjectModel	initWithContentsOfURL:

Code Snippet
File Name /source code/foodintaketrackertests/basetests.m
Method - (NSManagedObjectModel *)managedObjectModel

```
....
102.        _managedObjectModel = [[NSManagedObjectModel alloc]
initWithContentsOfURL:modelURL];
```

Side Channel Data Leakage

Description

Side Channel Data Leakage\Path 6:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=6
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/loginviewcontroller.m	/source code/Helpers/logginghelper.m
Line	111	44
Object	_txtPassword	NSLog

Code Snippet
File Name /source code/foodintaketracker/controller/loginviewcontroller.m
Method @implementation LoginViewController

```
....
111.  @implementation LoginViewController
```

▼

File Name /source code/Helpers/logginghelper.m
Method +(void)logMethodEntrance:(NSString *)methodName paramNames:(NSArray *)paramNames params:(NSArray *)params {

```
....
44.              NSLog(@"%@]]", log);
```

Side Channel Data Leakage\Path 7:

Severity	Medium

Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=7
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/loginviewcontroller.h	/source code/foodintaketracker/controller/loginviewcontroller.h
Line	92	92
Object	txtPassword	txtPassword

Code Snippet

File Name	/source code/foodintaketracker/controller/loginviewcontroller.h
Method	@property (weak, nonatomic) IBOutlet UITextField *txtPassword;

```
....
92.   @property (weak, nonatomic) IBOutlet UITextField *txtPassword;
```

Format String Attack

Description

Format String Attack\Path 18:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=18
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/addfoodviewcontroller.m	/source code/Helpers/logginghelper.m
Line	200	44
Object	text	NSLog

Code Snippet

File Name	/source code/foodintaketracker/controller/addfoodviewcontroller.m
Method	- (void)textFieldDidChange:(NSNotification *)notification {

```
....
200.          NSString *searchText = textField.text;
```

File Name	/source code/Helpers/logginghelper.m
Method	+(void)logMethodEntrance:(NSString *)methodName paramNames:(NSArray *)paramNames params:(NSArray *)params {

```
....
44.                NSLog(@"%@]]", log);
```

Format String Attack\Path 19:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=19
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/addfoodviewcontroller.m	/source code/Helpers/logginghelper.m
Line	200	53
Object	text	NSLog

Code Snippet

File Name	/source code/foodintaketracker/controller/addfoodviewcontroller.m
Method	- (void)textFieldDidChange:(NSNotification *)notification {

```
....
200.            NSString *searchText = textField.text;
```

▾

File Name	/source code/Helpers/logginghelper.m
Method	+(void)logMethodExit:(NSString *)methodName returnValue:(id)value {

```
....
53.                NSLog(@"[Output parameter %@]", value);
```

Cut And Paste Leakage

Description

Cut And Paste Leakage\Path 3:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=3
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/loginviewcontroller.h	/source code/foodintaketracker/controller/loginviewcontroller.h
Line	92	92

Object	txtPassword	txtPassword

Code Snippet
File Name /source code/foodintaketracker/controller/loginviewcontroller.h
Method @property (weak, nonatomic) IBOutlet UITextField *txtPassword;

```
....
92.   @property (weak, nonatomic) IBOutlet UITextField *txtPassword;
```

Insecure Data Storage

Description

Insecure Data Storage\Path 4:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=4
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/loginviewcontroller.m	/source code/Helpers/logginghelper.m
Line	111	44
Object	_txtPassword	NSLog

Code Snippet
File Name /source code/foodintaketracker/controller/loginviewcontroller.m
Method @implementation LoginViewController

```
....
111.   @implementation LoginViewController
```

File Name /source code/Helpers/logginghelper.m
Method +(void)logMethodEntrance:(NSString *)methodName paramNames:(NSArray *)paramNames params:(NSArray *)params {

```
....
44.             NSLog(@"%@]]", log);
```

Screen Caching

Description

Screen Caching\Path 22:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=22
Result Comment	

Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/loginviewcontroller.h	/source code/foodintaketracker/controller/loginviewcontroller.h
Line	92	92
Object	txtPassword	txtPassword

Code Snippet

File Name	/source code/foodintaketracker/controller/loginviewcontroller.h
Method	@property (weak, nonatomic) IBOutlet UITextField *txtPassword;

```
....
92.   @property (weak, nonatomic) IBOutlet UITextField *txtPassword;
```

Autocorrection Keystroke Logging

Description

Autocorrection Keystroke Logging\Path 131:

Severity	Medium
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=131
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/loginviewcontroller.h	/source code/foodintaketracker/controller/loginviewcontroller.h
Line	92	92
Object	txtPassword	txtPassword

Code Snippet

File Name	/source code/foodintaketracker/controller/loginviewcontroller.h
Method	@property (weak, nonatomic) IBOutlet UITextField *txtPassword;

```
....
92.   @property (weak, nonatomic) IBOutlet UITextField *txtPassword;
```

Unscrubbed Secret

Description

Unscrubbed Secret\Path 72:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=72
Result Comment	
Status	New

	Source	Destination
File	/source code/Services/basecommunicationdataservice.m	/source code/Services/basecommunicationdataservice.m
Line	30	30
Object	_sharedFileServerPassword	_sharedFileServerPassword

Code Snippet
File Name /source code/Services/basecommunicationdataservice.m
Method @synthesize sharedFileServerPassword = _sharedFileServerPassword;

```
....
30.  @synthesize sharedFileServerPassword = _sharedFileServerPassword;
```

Unscrubbed Secret\Path 73:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=73
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychainquery.m	/source code/library/postgresql/sskeychain/sskeychainquery.m
Line	17	17
Object	_passwordData	_passwordData

Code Snippet
File Name /source code/library/postgresql/sskeychain/sskeychainquery.m
Method @synthesize passwordData = _passwordData;

```
....
17.  @synthesize passwordData = _passwordData;
```

Unscrubbed Secret\Path 74:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=74
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskey	/source code/library/postgresql/sskeychain/sskey

	chainquery.m	chainquery.m
Line	17	17
Object	_passwordObject	_passwordObject

Code Snippet
File Name /source code/library/postgresql/sskeychain/sskeychainquery.m
Method @synthesize passwordData = _passwordData;

```
....
17.  @synthesize passwordData = _passwordData;
```

Unscrubbed Secret\Path 75:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=75
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychainquery.m	/source code/library/postgresql/sskeychain/sskeychainquery.m
Line	17	17
Object	_password	_password

Code Snippet
File Name /source code/library/postgresql/sskeychain/sskeychainquery.m
Method @synthesize passwordData = _passwordData;

```
....
17.  @synthesize passwordData = _passwordData;
```

Unscrubbed Secret\Path 76:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=76
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychainquery.m	/source code/library/postgresql/sskeychain/sskeychainquery.m
Line	17	17
Object	passwordData	passwordData

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychainquery.m
Method	@synthesize passwordData = _passwordData;

```
....
17.   @synthesize passwordData = _passwordData;
```

Unscrubbed Secret\Path 77:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=77
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychainquery.h	/source code/library/postgresql/sskeychain/sskeychainquery.h
Line	45	45
Object	passwordData	passwordData

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychainquery.h
Method	@property (nonatomic, copy) NSData *passwordData;

```
....
45.   @property (nonatomic, copy) NSData *passwordData;
```

Unscrubbed Secret\Path 78:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=78
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychainquery.m	/source code/library/postgresql/sskeychain/sskeychainquery.m
Line	17	17
Object	setPassworddata	setPassworddata

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychainquery.m
Method	@synthesize passwordData = _passwordData;

```
....
17.   @synthesize passwordData = _passwordData;
```

Unscrubbed Secret\Path 79:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=79
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychainquery.m	/source code/library/postgresql/sskeychain/sskeychainquery.m
Line	148	148
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychainquery.m
Method	- (void)setPassword:(NSString *)password {

```
....
148.   - (void)setPassword:(NSString *)password {
```

Unscrubbed Secret\Path 80:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=80
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychain.m	/source code/library/postgresql/sskeychain/sskeychain.m
Line	31	31
Object	passwordForService:account:error:	passwordForService:account:error:

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychain.m
Method	+ (NSString *)passwordForService:(NSString *)serviceName account:(NSString *)account error:(NSError *__autoreleasing *)error {

```
....
31.   + (NSString *)passwordForService:(NSString *)serviceName 
account:(NSString *)account error:(NSError *__autoreleasing *)error {
```

Unscrubbed Secret\Path 81:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=81
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychain.h	/source code/library/postgresql/sskeychain/sskeychain.h
Line	77	77
Object	passwordForService:account:error:	passwordForService:account:error:

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychain.h
Method	+ (NSString *)passwordForService:(NSString *)serviceName account:(NSString *)account error:(NSError **)error;

```
....
77.   + (NSString *)passwordForService:(NSString *)serviceName 
account:(NSString *)account error:(NSError **)error;
```

Unscrubbed Secret\Path 82:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=82
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychainquery.m	/source code/library/postgresql/sskeychain/sskeychainquery.m
Line	153	153
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychainquery.m
Method	- (NSString *)password {

```
....
153.    - (NSString *)password {
```

Unscrubbed Secret\Path 83:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=83
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychainquery.h	/source code/library/postgresql/sskeychain/sskeychainquery.h
Line	57	57
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychainquery.h
Method	@property (nonatomic, copy) NSString *password;

```
....
57.   @property (nonatomic, copy) NSString *password;
```

Unscrubbed Secret\Path 84:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=84
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychain.m	/source code/library/postgresql/sskeychain/sskeychain.m
Line	45	45
Object	deletePasswordForService:account:error:	deletePasswordForService:account:error:

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychain.m
Method	+ (BOOL)deletePasswordForService:(NSString *)serviceName account:(NSString *)account error:(NSError *__autoreleasing *)error {

```
....
45.   + (BOOL)deletePasswordForService:(NSString *)serviceName
account:(NSString *)account error:(NSError *__autoreleasing *)error {
```

Unscrubbed Secret\Path 85:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=85
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychain.h	/source code/library/postgresql/sskeychain/sskeychain.h
Line	90	90
Object	deletePasswordForService:account:error:	deletePasswordForService:account:error:

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychain.h
Method	+ (BOOL)deletePasswordForService:(NSString *)serviceName account:(NSString *)account error:(NSError **)error;

```
....
90.   + (BOOL)deletePasswordForService:(NSString *)serviceName
account:(NSString *)account error:(NSError **)error;
```

Unscrubbed Secret\Path 86:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=86
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychain.m	/source code/library/postgresql/sskeychain/sskeychain.m
Line	53	53
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychain.m
Method	+ (BOOL)setPassword:(NSString *)password forService:(NSString *)serviceName account:(NSString *)account {

```
....
53.   + (BOOL)setPassword:(NSString *)password forService:(NSString
*)serviceName account:(NSString *)account {
```

Unscrubbed Secret\Path 87:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=87
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychain.m	/source code/library/postgresql/sskeychain/sskeychain.m
Line	58	58
Object	setPassword:forService:account:error:	setPassword:forService:account:error:

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychain.m
Method	+ (BOOL)setPassword:(NSString *)password forService:(NSString *)serviceName account:(NSString *)account error:(NSError *__autoreleasing *)error {

```
....
58.   + (BOOL)setPassword:(NSString *)password forService:(NSString
*)serviceName account:(NSString *)account error:(NSError
*__autoreleasing *)error {
```

Unscrubbed Secret\Path 88:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=88
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychain.h	/source code/library/postgresql/sskeychain/sskeychain.h
Line	105	105
Object	setPassword:forService:account:error:	setPassword:forService:account:error:

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychain.h
Method	+ (BOOL)setPassword:(NSString *)password forService:(NSString *)serviceName account:(NSString *)account error:(NSError **)error;

```
....
105.   + (BOOL)setPassword:(NSString *)password forService:(NSString
*)serviceName account:(NSString *)account error:(NSError **)error;
```

Unscrubbed Secret\Path 89:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=89
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychain.m	/source code/library/postgresql/sskeychain/sskeychain.m
Line	58	58
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychain.m
Method	+ (BOOL)setPassword:(NSString *)password forService:(NSString *)serviceName account:(NSString *)account error:(NSError *__autoreleasing *)error {

```
....
58.   + (BOOL)setPassword:(NSString *)password forService:(NSString
*)serviceName account:(NSString *)account error:(NSError
*__autoreleasing *)error {
```

Unscrubbed Secret\Path 90:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=90
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychainquery.m	/source code/library/postgresql/sskeychain/sskeychainquery.m
Line	17	17
Object	setPassword	setPassword

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychainquery.m
Method	@synthesize passwordData = _passwordData;

```
....
17.  @synthesize passwordData = _passwordData;
```

Unscrubbed Secret\Path 91:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=91
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.h	/source code/library/postgresql/pgclientkit/pgpasswordstore.h
Line	2	2
Object	iPGPasswordStore	iPGPasswordStore

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgpasswordstore.h
Method	@interface PGPasswordStore : NSObject {

```
....
2.  @interface PGPasswordStore : NSObject {
```

Unscrubbed Secret\Path 92:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=92
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.m	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Line	6	6
Object	PGPasswordStore	PGPasswordStore

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Method	@implementation PGPasswordStore

```
....
6.  @implementation PGPasswordStore
```

Unscrubbed Secret\Path 93:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=93
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.m	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Line	61	61
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Method	-(NSString*)passwordForURL:(NSURL*)url error:(NSError**)error {

```
....
61.    NSString* password = [url password];
```

Unscrubbed Secret\Path 94:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=94
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.m	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Line	59	59
Object	passwordForURL:error:	passwordForURL:error:

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Method	-(NSString*)passwordForURL:(NSURL*)url error:(NSError**)error {

```
....
59.   -(NSString* )passwordForURL:(NSURL* )url error:(NSError** )error {
```

Unscrubbed Secret\Path 95:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=95

Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.h	/source code/library/postgresql/pgclientkit/pgpasswordstore.h
Line	11	11
Object	passwordForURL:error:	passwordForURL:error:

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgpasswordstore.h
Method	-(NSString*)passwordForURL:(NSURL*)url error:(NSError**)error;

```
....
11.  -(NSString* )passwordForURL:(NSURL* )url error:(NSError** )error;
```

Unscrubbed Secret\Path 96:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=96
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.m	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Line	87	87
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Method	-(BOOL)setPassword:(NSString*)password forURL:(NSURL*)url saveToKeychain:(BOOL)saveToKeychain {

```
....
87.  -(BOOL)setPassword:(NSString* )password forURL:(NSURL* )url
saveToKeychain:(BOOL)saveToKeychain {
```

Unscrubbed Secret\Path 97:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=97
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.m	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Line	91	91
Object	setPassword:forURL:saveToKeychain:error:	setPassword:forURL:saveToKeychain:error:

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Method	-(BOOL)setPassword:(NSString*)password forURL:(NSURL*)url saveToKeychain:(BOOL)saveToKeychain error:(NSError**)error {

```
....
91.   -(BOOL)setPassword:(NSString* )password forURL:(NSURL* )url
saveToKeychain:(BOOL)saveToKeychain error:(NSError** )error {
```

Unscrubbed Secret\Path 98:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=98
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.h	/source code/library/postgresql/pgclientkit/pgpasswordstore.h
Line	13	13
Object	setPassword:forURL:saveToKeychain:error:	setPassword:forURL:saveToKeychain:error:

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgpasswordstore.h
Method	-(BOOL)setPassword:(NSString*)password forURL:(NSURL*)url saveToKeychain:(BOOL)saveToKeychain error:(NSError**)error;

```
....
13.   -(BOOL)setPassword:(NSString* )password forURL:(NSURL* )url
saveToKeychain:(BOOL)saveToKeychain error:(NSError** )error;
```

Unscrubbed Secret\Path 99:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=99
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.m	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Line	91	91
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgpasswordstore.m
Method	-(BOOL)setPassword:(NSString*)password forURL:(NSURL*)url saveToKeychain:(BOOL)saveToKeychain error:(NSError**)error {

```
....
91.   -(BOOL)setPassword:(NSString* )password forURL:(NSURL* )url
saveToKeychain:(BOOL)saveToKeychain error:(NSError** )error {
```

Unscrubbed Secret\Path 100:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=100
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/include/libpq-fe.h	/source code/library/postgresql/include/libpq-fe.h
Line	308	308
Object	PQconnectionNeedsPassword	PQconnectionNeedsPassword

Code Snippet

File Name	/source code/library/postgresql/include/libpq-fe.h
Method	extern int PQconnectionNeedsPassword(const PGconn *conn);

```
....
308.  extern int  PQconnectionNeedsPassword(const PGconn *conn);
```

Unscrubbed Secret\Path 101:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=101
Result Comment	
Status	New

	Source	Destination
File	/source	/source

	code/library/postgresql/include/libpq-fe.h	code/library/postgresql/include/libpq-fe.h
Line	309	309
Object	PQconnectionUsedPassword	PQconnectionUsedPassword

Code Snippet

File Name	/source code/library/postgresql/include/libpq-fe.h
Method	extern int PQconnectionUsedPassword(const PGconn *conn);

```
....
309.  extern int  PQconnectionUsedPassword(const PGconn *conn);
```

Unscrubbed Secret\Path 102:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=102
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Line	52	52
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Method	+(NSString*)_pg_urlencode_password:(NSString*)password {

```
....
52.  +(NSString* )_pg_urlencode_password:(NSString* )password {
```

Unscrubbed Secret\Path 103:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=103
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Line	124	124

Object	password	password

Code Snippet
File Name /source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Method +(id)URLWithHost:(NSString*)host ssl:(BOOL)ssl username:(NSString*)username password:(NSString *) password database:(NSString*)database params:(NSDictionary*)params {

```
....
124.   +(id)URLWithHost:(NSString* )host ssl:(BOOL)ssl
username:(NSString* )username password:(NSString *) password
database:(NSString* )database params:(NSDictionary* )params {
```

Unscrubbed Secret\Path 104:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=104
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/nsurl +pgadditions.m	/source code/library/postgresql/pgclientkit/nsurl +pgadditions.m
Line	128	128
Object	password	password

Code Snippet
File Name /source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Method +(id)URLWithHost:(NSString*)host port:(NSUInteger)port ssl:(BOOL)ssl username:(NSString*)username password:(NSString *) password database:(NSString*)database params:(NSDictionary*)params {

```
....
128.   +(id)URLWithHost:(NSString* )host port:(NSUInteger)port
ssl:(BOOL)ssl username:(NSString* )username password:(NSString *)
password database:(NSString* )database params:(NSDictionary* )params {
```

Unscrubbed Secret\Path 105:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=105
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/nsurl	/source code/library/postgresql/pgclientkit/nsurl

	+pgadditions.m	+pgadditions.m
Line	151	151
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Method	-(id)initWithHost:(NSString*)host port:(NSUInteger)port ssl:(BOOL)ssl username:(NSString*)username password:(NSString *)password database:(NSString *)database params:(NSDictionary *)params {

```
....
151.  -(id)initWithHost:(NSString* )host port:(NSUInteger)port
ssl:(BOOL)ssl username:(NSString* )username password:(NSString
*)password database:(NSString *)database params:(NSDictionary *)params {
```

Unscrubbed Secret\Path 106:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=106
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/nsurl +pgadditions.m	/source code/library/postgresql/pgclientkit/nsurl +pgadditions.m
Line	156	156
Object	passwordenc	passwordenc

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Method	-(id)initWithHost:(NSString*)host port:(NSUInteger)port ssl:(BOOL)ssl username:(NSString*)username password:(NSString *)password database:(NSString *)database params:(NSDictionary *)params {

```
....
156.       NSString* passwordenc = [NSURL
_pg_urlencode_password:[password
stringByTrimmingCharactersInSet:[NSCharacterSet
whitespaceAndNewlineCharacterSet]]];
```

Unscrubbed Secret\Path 107:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=107
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Line	52	52
Object	_pg_urlencode_password:	_pg_urlencode_password:

Code Snippet
File Name /source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Method +(NSString*)_pg_urlencode_password:(NSString*)password {

```
....
52.   +(NSString* )_pg_urlencode_password:(NSString* )password {
```

Unscrubbed Secret\Path 108:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=108
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Line	164	164
Object	password	password

Code Snippet
File Name /source code/library/postgresql/pgclientkit/nsurl+pgadditions.m
Method -(id)initWithHost:(NSString*)host ssl:(BOOL)ssl username:(NSString*)username password:(NSString *)password database:(NSString *)database params:(NSDictionary *)params {

```
....
164.   -(id)initWithHost:(NSString* )host ssl:(BOOL)ssl
username:(NSString* )username password:(NSString *)password
database:(NSString *)database params:(NSDictionary *)params {
```

Unscrubbed Secret\Path 109:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=109
Result Comment	
Status	New

	Source	Destination

File	/source code/Services/basecommunicationdataservice.m	/source code/Services/basecommunicationdataservice.m
Line	30	30
Object	sharedFileServerPassword	sharedFileServerPassword

Code Snippet
File Name /source code/Services/basecommunicationdataservice.m
Method @synthesize sharedFileServerPassword = _sharedFileServerPassword;

```
....
30.  @synthesize sharedFileServerPassword = _sharedFileServerPassword;
```

Unscrubbed Secret\Path 110:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=110
Result Comment	
Status	New

	Source	Destination
File	/source code/Services/basecommunicationdataservice.h	/source code/Services/basecommunicationdataservice.h
Line	52	52
Object	sharedFileServerPassword	sharedFileServerPassword

Code Snippet
File Name /source code/Services/basecommunicationdataservice.h
Method @property (nonatomic, readonly, strong) NSString *sharedFileServerPassword;

```
....
52.  @property (nonatomic, readonly, strong) NSString
*sharedFileServerPassword;
```

Unscrubbed Secret\Path 111:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=111
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/loginviewcontroller.m	/source code/foodintaketracker/controller/loginviewcontroller.m

Line	111	111
Object	_txtPassword	_txtPassword

Code Snippet

File Name	/source code/foodintaketracker/controller/loginviewcontroller.m
Method	@implementation LoginViewController

```
....
111.   @implementation LoginViewController
```

Unscrubbed Secret\Path 112:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=112
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/loginviewcontroller.m	/source code/foodintaketracker/controller/loginviewcontroller.m
Line	111	111
Object	txtPassword	txtPassword

Code Snippet

File Name	/source code/foodintaketracker/controller/loginviewcontroller.m
Method	@implementation LoginViewController

```
....
111.   @implementation LoginViewController
```

Unscrubbed Secret\Path 113:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=113
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/common/pgcoredata.m	/source code/foodintaketracker/common/pgcoredata.m
Line	48	48
Object	password	password

Code Snippet

File Name	/source code/foodintaketracker/common/pgcoredata.m
Method	- (BOOL)connect {

```
....
48.       NSString *password = [standardUserDefaults
objectForKey:@"password_preference"];
```

Unscrubbed Secret\Path 114:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=114
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychainquery.h	/source code/library/postgresql/sskeychain/sskeychainquery.h
Line	51	51
Object	passwordObject	passwordObject

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychainquery.h
Method	@property (nonatomic, copy) id<NSCoding> passwordObject;

```
....
51.  @property (nonatomic, copy) id<NSCoding> passwordObject;
```

Unscrubbed Secret\Path 115:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=115
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychain.h	/source code/library/postgresql/sskeychain/sskeychain.h
Line	104	104
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/sskeychain/sskeychain.h
Method	+ (BOOL)setPassword:(NSString *)password forService:(NSString *)serviceName account:(NSString *)account;

```
....
104.   + (BOOL)setPassword:(NSString *)password forService:(NSString
*)serviceName account:(NSString *)account;
```

Unscrubbed Secret\Path 116:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=116
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/sskeychain/sskeychain.h	/source code/library/postgresql/sskeychain/sskeychain.h
Line	105	105
Object	password	password

Code Snippet

File Name /source code/library/postgresql/sskeychain/sskeychain.h

Method + (BOOL)setPassword:(NSString *)password forService:(NSString *)serviceName account:(NSString *)account error:(NSError **)error;

```
....
105.   + (BOOL)setPassword:(NSString *)password forService:(NSString
*)serviceName account:(NSString *)account error:(NSError **)error;
```

Unscrubbed Secret\Path 117:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=117
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.h	/source code/library/postgresql/pgclientkit/pgpasswordstore.h
Line	12	12
Object	password	password

Code Snippet

File Name /source code/library/postgresql/pgclientkit/pgpasswordstore.h

Method -(BOOL)setPassword:(NSString*)password forURL:(NSURL*)url saveToKeychain:(BOOL)saveToKeychain;

```
....
12.   -(BOOL)setPassword:(NSString* )password forURL:(NSURL* )url
saveToKeychain:(BOOL)saveToKeychain;
```

Unscrubbed Secret\Path 118:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=118
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgpasswordstore.h	/source code/library/postgresql/pgclientkit/pgpasswordstore.h
Line	13	13
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgpasswordstore.h
Method	-(BOOL)setPassword:(NSString*)password forURL:(NSURL*)url saveToKeychain:(BOOL)saveToKeychain error:(NSError**)error;

```
....
13.   -(BOOL)setPassword:(NSString* )password forURL:(NSURL* )url
saveToKeychain:(BOOL)saveToKeychain error:(NSError** )error;
```

Unscrubbed Secret\Path 119:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=119
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgclientkit.h	/source code/library/postgresql/pgclientkit/pgclientkit.h
Line	35	35
Object	PGPasswordStore	PGPasswordStore

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgclientkit.h
Method	@class PGPasswordStore;

```
....
35.  @class PGPasswordStore;
```

Unscrubbed Secret\Path 120:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=120
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.h	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.h
Line	62	62
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.h
Method	+(id)URLWithHost:(NSString*)host ssl:(BOOL)ssl username:(NSString*)username password:(NSString*)password

```
....
62.  +(id)URLWithHost:(NSString* )host ssl:(BOOL)ssl username:(NSString*
)username password:(NSString* )password
```

Unscrubbed Secret\Path 121:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=121
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.h	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.h
Line	96	96
Object	password	password

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/nsurl+pgadditions.h
Method	+(id)URLWithHost:(NSString*)host port:(NSUInteger)port ssl:(BOOL)ssl username:(NSString*)username password:(NSString*)password

```
....
96.   +(id)URLWithHost:(NSString* )host port:(NSUInteger)port
ssl:(BOOL)ssl username:(NSString* )username password:(NSString*
)password
```

Jailbrake File Referenced By Name

[Description]()

Jailbrake File Referenced By Name\Path 28:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=28
Result Comment	
Status	New

	Source	Destination
File	/source code/Services/synchronizationserviceimpl.m	/source code/Services/synchronizationserviceimpl.m
Line	314	314
Object	writeToFile:options:error:	writeToFile:options:error:

Code Snippet

File Name	/source code/Services/synchronizationserviceimpl.m
Method	-(BOOL)synchronize:(NSError **)error{

```
....
314.                if (![data writeToFile:localDataFile
options:NSDataWritingAtomic error:&e]) {
```

Jailbrake File Referenced By Name\Path 29:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=29
Result Comment	
Status	New

	Source	Destination
File	/source code/Services/dataupdateserviceimpl.m	/source code/Services/dataupdateserviceimpl.m
Line	194	194
Object	writeToFile:options:error:	writeToFile:options:error:

Code Snippet

File Name	/source code/Services/dataupdateserviceimpl.m
Method	-(BOOL)update:(NSError **)error force:(BOOL)force {

```
....
194.                      [data writeToFile:localDataFile 
options:NSDataWritingAtomic error:&e];
```

Jailbrake File Referenced By Name\Path 30:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=30
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/common/Helper.m	/source code/foodintaketracker/common/Helper.m
Line	75	75
Object	writeToFile:atomically:	writeToFile:atomically:

Code Snippet
File Name /source code/foodintaketracker/common/Helper.m
Method + (NSString *)saveImage:(NSData *)data {

```
....
75.         [data writeToFile:filePath atomically:YES];
```

Jailbrake File Referenced By Name\Path 31:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=31
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/common/Helper.m	/source code/foodintaketracker/common/Helper.m
Line	120	120
Object	writeToFile:atomically:	writeToFile:atomically:

Code Snippet
File Name /source code/foodintaketracker/common/Helper.m
Method + (NSString *)saveVoiceRecording:(NSData *)data {

```
....
120.         [data writeToFile:filePath atomically:YES];
```

Jailbrake File Referenced By Name\Path 32:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=32
Result Comment	
Status	New

	Source	Destination
File	/source code/Services/synchronizationserviceimpl.m	/source code/Services/synchronizationserviceimpl.m
Line	347	347
Object	dataWithContentsOfFile:	dataWithContentsOfFile:

Code Snippet
File Name /source code/Services/synchronizationserviceimpl.m
Method - (BOOL)saveMedia:(SynchronizableModel *) object {

```
....
347.          NSData *data = [NSData dataWithContentsOfFile:localPath];
```

Jailbrake File Referenced By Name\Path 33:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=33
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/bnappstats.m	/source code/library/bnpiechart/bnappstats.m
Line	53	53
Object	stringWithContentsOfFile:usedEncoding:error:	stringWithContentsOfFile:usedEncoding:error:

Code Snippet
File Name /source code/library/bnpiechart/bnappstats.m
Method + (int)loadIntWithIndex:(int)index {

```
....
53.    NSString *fileContents = [NSString stringWithContentsOfFile:path
usedEncoding:&enc error:NULL];
```

Jailbrake File Referenced By Name\Path 34:

Severity	Low
Result State	To Verify

Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=34
Result Comment	
Status	New

	Source	Destination
File	/source code/issfoodintaketrackertests/synchronizationservicetests.m	/source code/issfoodintaketrackertests/synchronizationservicetests.m
Line	35	35
Object	dataWithContentsOfFile:	dataWithContentsOfFile:

Code Snippet
File Name /source code/issfoodintaketrackertests/synchronizationservicetests.m
Method - (BOOL)copyLocalFile:(NSString *)localFile toSMBPath:(NSString *)smbPath error:(NSError **)error {

```
....
35.       NSData *data = [NSData dataWithContentsOfFile:localFile];
```

Jailbrake File Referenced By Name\Path 35:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=35
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/synchronizationservicetests.m	/source code/foodintaketrackertests/synchronizationservicetests.m
Line	35	35
Object	dataWithContentsOfFile:	dataWithContentsOfFile:

Code Snippet
File Name /source code/foodintaketrackertests/synchronizationservicetests.m
Method - (BOOL)copyLocalFile:(NSString *)localFile toSMBPath:(NSString *)smbPath error:(NSError **)error {

```
....
35.       NSData *data = [NSData dataWithContentsOfFile:localFile];
```

Jailbrake File Referenced By Name\Path 36:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=36
Result Comment	

Status	New

	Source	Destination
File	/source code/issfoodintaketrackertests/dataupdateservicetests.m	/source code/issfoodintaketrackertests/dataupdateservicetests.m
Line	37	37
Object	dataWithContentsOfFile:	dataWithContentsOfFile:

Code Snippet
File Name /source code/issfoodintaketrackertests/dataupdateservicetests.m
Method - (BOOL)copyLocalFile:(NSString *)localFile toSMBPath:(NSString *)smbPath error:(NSError **)error {

```
....
37.       NSData *data = [NSData dataWithContentsOfFile:localFile];
```

Jailbrake File Referenced By Name\Path 37:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=37
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/dataupdateservicetests.m	/source code/foodintaketrackertests/dataupdateservicetests.m
Line	37	37
Object	dataWithContentsOfFile:	dataWithContentsOfFile:

Code Snippet
File Name /source code/foodintaketrackertests/dataupdateservicetests.m
Method - (BOOL)copyLocalFile:(NSString *)localFile toSMBPath:(NSString *)smbPath error:(NSError **)error {

```
....
37.       NSData *data = [NSData dataWithContentsOfFile:localFile];
```

Jailbrake File Referenced By Name\Path 38:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=38
Result Comment	
Status	New

	Source	Destination

File	/source code/issfoodintaketrackertests/basetests.m	/source code/issfoodintaketrackertests/basetests.m
Line	38	38
Object	dictionaryWithContentsOfFile:	dictionaryWithContentsOfFile:

Code Snippet
File Name /source code/issfoodintaketrackertests/basetests.m
Method -(void)setUp

```
....
38.       self.configurations = [NSMutableDictionary
dictionaryWithContentsOfFile:path];
```

Jailbrake File Referenced By Name\Path 39:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=39
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/basetests.m	/source code/foodintaketrackertests/basetests.m
Line	38	38
Object	dictionaryWithContentsOfFile:	dictionaryWithContentsOfFile:

Code Snippet
File Name /source code/foodintaketrackertests/basetests.m
Method -(void)setUp

```
....
38.       self.configurations = [NSMutableDictionary
dictionaryWithContentsOfFile:path];
```

Jailbrake File Referenced By Name\Path 40:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=40
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/DBHelper.m	/source code/Helpers/DBHelper.m
Line	177	177
Object	dictionaryWithContentsOfFile:	dictionaryWithContentsOfFile:

Code Snippet	
File Name	/source code/Helpers/DBHelper.m
Method	dispatch_once(&onceToken, ^{

```
....
177.          NSDictionary *configuration = [NSDictionary
dictionaryWithContentsOfFile:configBundle];
```

Jailbrake File Referenced By Name\Path 41:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=41
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/common/Helper.m	/source code/foodintaketracker/common/Helper.m
Line	196	196
Object	imageWithContentsOfFile:	imageWithContentsOfFile:

Code Snippet	
File Name	/source code/foodintaketracker/common/Helper.m
Method	+(UIImage *)loadImage:(NSString *)imagePath {

```
....
196.      UIImage *image = [UIImage imageWithContentsOfFile:filePath];
```

Jailbrake File Referenced By Name\Path 42:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=42
Result Comment	
Status	New

	Source	Destination
File	/source code/issfoodintaketrackertests/basetests.m	/source code/issfoodintaketrackertests/basetests.m
Line	102	102
Object	initWithContentsOfURL:	initWithContentsOfURL:

Code Snippet	
File Name	/source code/issfoodintaketrackertests/basetests.m
Method	- (NSManagedObjectModel *)managedObjectModel

```
....
102.        _managedObjectModel = [[NSManagedObjectModel alloc] 
initWithContentsOfURL:modelURL];
```

Jailbrake File Referenced By Name\Path 43:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=43
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/basetests.m	/source code/foodintaketrackertests/basetests.m
Line	102	102
Object	initWithContentsOfURL:	initWithContentsOfURL:

Code Snippet
File Name /source code/foodintaketrackertests/basetests.m
Method - (NSManagedObjectModel *)managedObjectModel

```
....
102.        _managedObjectModel = [[NSManagedObjectModel alloc] 
initWithContentsOfURL:modelURL];
```

Jailbrake File Referenced By Name\Path 44:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=44
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/helpsettingviewcontroller.m	/source code/foodintaketracker/controller/helpsettingviewcontroller.m
Line	320	320
Object	stringWithContentsOfURL:encoding:error:	stringWithContentsOfURL:encoding:error:

Code Snippet
File Name /source code/foodintaketracker/controller/helpsettingviewcontroller.m
Method - (void) loadHelpDetails {

```
....
320.        NSString *html = [NSString stringWithContentsOfURL:url
encoding:NSUTF8StringEncoding error:nil];
```

Jailbrake File Referenced By Name\Path 45:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=45
Result Comment	
Status	New

	Source	Destination
File	/source code/issfoodintaketrackertests/synchronizationservicetests.m	/source code/issfoodintaketrackertests/synchronizationservicetests.m
Line	195	195
Object	copyItemAtPath:toPath:error:	copyItemAtPath:toPath:error:

Code Snippet
File Name /source code/issfoodintaketrackertests/synchronizationservicetests.m
Method - (void)setUp {

```
....
195.            [[NSFileManager defaultManager]
copyItemAtPath:[localFolder stringByAppendingPathComponent:file]
```

Jailbrake File Referenced By Name\Path 46:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=46
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/synchronizationservicetests.m	/source code/foodintaketrackertests/synchronizationservicetests.m
Line	185	185
Object	copyItemAtPath:toPath:error:	copyItemAtPath:toPath:error:

Code Snippet
File Name /source code/foodintaketrackertests/synchronizationservicetests.m
Method - (void)setUp {

```
....
185.          [[NSFileManager defaultManager]
copyItemAtPath:[localFolder stringByAppendingPathComponent:file]
```

Jailbrake File Referenced By Name\Path 47:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=47
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/DBHelper.m	/source code/Helpers/DBHelper.m
Line	204	204
Object	copyItemAtPath:toPath:error:	copyItemAtPath:toPath:error:

Code Snippet

File Name	/source code/Helpers/DBHelper.m
Method	dispatch_once(&onceToken, ^{

```
....
204.                      [[NSFileManager defaultManager]
copyItemAtPath:[localFolder stringByAppendingPathComponent:file]
```

Jailbrake File Referenced By Name\Path 48:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=48
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/DBHelper.m	/source code/Helpers/DBHelper.m
Line	210	210
Object	copyItemAtPath:toPath:error:	copyItemAtPath:toPath:error:

Code Snippet

File Name	/source code/Helpers/DBHelper.m
Method	dispatch_once(&onceToken, ^{

```
....
210.                      [[NSFileManager defaultManager]
copyItemAtPath:sqlFile
```

Jailbrake File Referenced By Name\Path 49:

Severity	Low

Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=49
Result Comment	
Status	New

	Source	Destination
File	/source code/issfoodintaketrackertests/speechrecognitionservicetests.m	/source code/issfoodintaketrackertests/speechrecognitionservicetests.m
Line	85	85
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet

File Name	/source code/issfoodintaketrackertests/speechrecognitionservicetests.m
Method	-(void)tearDown {

```
....
85.       if ([fm fileExistsAtPath:storeURL.path]) {
```

Jailbrake File Referenced By Name\Path 50:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=50
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/speechrecognitionservicetests.m	/source code/foodintaketrackertests/speechrecognitionservicetests.m
Line	83	83
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet

File Name	/source code/foodintaketrackertests/speechrecognitionservicetests.m
Method	-(void)tearDown {

```
....
83.       if ([fm fileExistsAtPath:storeURL.path]) {
```

Jailbrake File Referenced By Name\Path 51:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=51
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/bnappstats.m	/source code/library/bnpiechart/bnappstats.m
Line	49	49
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet
File Name /source code/library/bnpiechart/bnappstats.m
Method + (int)loadIntWithIndex:(int)index {

```
....
49.    if (![[NSFileManager defaultManager] fileExistsAtPath:path]) {
```

Jailbrake File Referenced By Name\Path 52:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=52
Result Comment	
Status	New

	Source	Destination
File	/source code/issfoodintaketrackertests/synchronizationservicetests.m	/source code/issfoodintaketrackertests/synchronizationservicetests.m
Line	374	374
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet
File Name /source code/issfoodintaketrackertests/synchronizationservicetests.m
Method - (void)testSynchronizesynchronize {

```
....
374.          STAssertTrue([[NSFileManager defaultManager]
fileExistsAtPath:localDataFile], @"local file should exist.");
```

Jailbrake File Referenced By Name\Path 53:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=53
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/synchronizat	/source code/foodintaketrackertests/synchronizat

	ionservicetests.m	ionservicetests.m
Line	358	358
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet

File Name	/source code/foodintaketrackertests/synchronizationservicetests.m
Method	- (void)testSynchronizesynchronize {

```
....
358.            STAssertTrue([[NSFileManager defaultManager]
fileExistsAtPath:localDataFile], @"local file should exist.");
```

Jailbrake File Referenced By Name\Path 54:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=54
Result Comment	
Status	New

	Source	Destination
File	/source code/issfoodintaketrackertests/dataupdateservicetests.m	/source code/issfoodintaketrackertests/dataupdateservicetests.m
Line	236	236
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet

File Name	/source code/issfoodintaketrackertests/dataupdateservicetests.m
Method	- (void)testUpdate {

```
....
236.            STAssertTrue([[NSFileManager defaultManager]
fileExistsAtPath:localDataFile], @"local file should exist.");
```

Jailbrake File Referenced By Name\Path 55:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=55
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/dataupdateservicetests.m	/source code/foodintaketrackertests/dataupdateservicetests.m
Line	226	226

Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet
File Name /source code/foodintaketrackertests/dataupdateservicetests.m
Method - (void)testUpdate {

```
....
226.          STAssertTrue([[NSFileManager defaultManager]
fileExistsAtPath:localDataFile], @"local file should exist.");
```

Jailbrake File Referenced By Name\Path 56:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=56
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/DBHelper.m	/source code/Helpers/DBHelper.m
Line	182	182
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet
File Name /source code/Helpers/DBHelper.m
Method dispatch_once(&onceToken, ^{

```
....
182.              if([[NSFileManager defaultManager]
fileExistsAtPath:persistentStorePath]) {
```

Jailbrake File Referenced By Name\Path 57:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=57
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/DBHelper.m	/source code/Helpers/DBHelper.m
Line	195	195
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet
File Name /source code/Helpers/DBHelper.m
Method dispatch_once(&onceToken, ^{

```
....
195.          else  if(![[NSFileManager defaultManager]
fileExistsAtPath:persistentStorePath]) {
```

Jailbrake File Referenced By Name\Path 58:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=58
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/DBHelper.m	/source code/Helpers/DBHelper.m
Line	266	266
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet

File Name	/source code/Helpers/DBHelper.m
Method	+ (void)resetPersistentStore {

```
....
266.      if ([[NSFileManager defaultManager]
fileExistsAtPath:store.URL.path]) {
```

Jailbrake File Referenced By Name\Path 59:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=59
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/datahelper.m	/source code/Helpers/datahelper.m
Line	156	156
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet

File Name	/source code/Helpers/datahelper.m
Method	+ (NSString *)getAbsoulteLocalDirectory:(NSString *)localDirectory {

```
....
156.      if(![[NSFileManager defaultManager]
fileExistsAtPath:localDirFullPath]) {
```

Jailbrake File Referenced By Name\Path 60:

Severity	Low

Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=60
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/common/Helper.m	/source code/foodintaketracker/common/Helper.m
Line	62	62
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet

File Name	/source code/foodintaketracker/common/Helper.m
Method	+ (NSString *)saveImage:(NSData *)data {

```
....
62.       if (![[NSFileManager defaultManager]
fileExistsAtPath:additionalFileDirectory]) {
```

Jailbrake File Referenced By Name\Path 61:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=61
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/common/Helper.m	/source code/foodintaketracker/common/Helper.m
Line	110	110
Object	fileExistsAtPath:	fileExistsAtPath:

Code Snippet

File Name	/source code/foodintaketracker/common/Helper.m
Method	+ (NSString *)saveVoiceRecording:(NSData *)data {

```
....
110.       if (![[NSFileManager defaultManager]
fileExistsAtPath:additionalFileDirectory]) {
```

Use of Hardcoded Password

Description

Use of Hardcoded Password\Path 10:

Severity	Low
Result State	To Verify

Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=10
Result Comment	
Status	New

	Source	Destination
File	/source code/issfoodintaketrackertests/smbclienttests.m	/source code/issfoodintaketrackertests/smbclienttests.m
Line	413	413
Object	result	result

Code Snippet
File Name /source code/issfoodintaketrackertests/smbclienttests.m
Method - (void)testConnect {

```
....
413.       BOOL result = [self.smbClient
connect:self.configurations[@"SharedFileServerPath"]
```

Use of Hardcoded Password\Path 11:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=11
Result Comment	
Status	New

	Source	Destination
File	/source code/issfoodintaketrackertests/smbclienttests.m	/source code/issfoodintaketrackertests/smbclienttests.m
Line	428	428
Object	result	result

Code Snippet
File Name /source code/issfoodintaketrackertests/smbclienttests.m
Method - (void)testConnect_Failure{

```
....
428.       BOOL result = [self.smbClient
connect:self.configurations[@"SharedFileServerPath"]
```

Use of Hardcoded Password\Path 12:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=12
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/smbclienttests.m	/source code/foodintaketrackertests/smbclienttests.m
Line	411	411
Object	result	result

Code Snippet
File Name /source code/foodintaketrackertests/smbclienttests.m
Method - (void)testConnect {

```
....
411.      BOOL result = [self.smbClient
connect:self.configurations[@"SharedFileServerPath"]
```

Use of Hardcoded Password\Path 13:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=13
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketrackertests/smbclienttests.m	/source code/foodintaketrackertests/smbclienttests.m
Line	426	426
Object	result	result

Code Snippet
File Name /source code/foodintaketrackertests/smbclienttests.m
Method - (void)testConnect_Failure{

```
....
426.      BOOL result = [self.smbClient
connect:self.configurations[@"SharedFileServerPath"]
```

Jailbreak Unchecked File Operation Result Code

Description

Jailbreak Unchecked File Operation Result Code\Path 62:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=62
Result Comment	
Status	New

	Source	Destination
File	/source code/Services/dataupdateserviceimpl.m	/source code/Services/dataupdateserviceimpl.m
Line	194	194
Object	writeToFile:options:error:	writeToFile:options:error:

Code Snippet
File Name /source code/Services/dataupdateserviceimpl.m
Method -(BOOL)update:(NSError **)error force:(BOOL)force {

```
....
194.                    [data writeToFile:localDataFile
options:NSDataWritingAtomic error:&e];
```

Jailbreak Unchecked File Operation Result Code\Path 63:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=63
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/bnappstats.m	/source code/library/bnpiechart/bnappstats.m
Line	61	61
Object	writeToFile:atomically:encoding:error:	writeToFile:atomically:encoding:error:

Code Snippet
File Name /source code/library/bnpiechart/bnappstats.m
Method + (void)saveWithInits:(int)inits opens:(int)opens {

```
....
61.      [fileContents writeToFile:[self saveFile] atomically:YES
encoding:NSUTF8StringEncoding error:NULL];
```

Jailbreak Unchecked File Operation Result Code\Path 64:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=64
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/common/Helper.m	/source code/foodintaketracker/common/Helper.m

Line	75	75
Object	writeToFile:atomically:	writeToFile:atomically:

Code Snippet	
File Name	/source code/foodintaketracker/common/Helper.m
Method	+ (NSString *)saveImage:(NSData *)data {

```
....
75.       [data writeToFile:filePath atomically:YES];
```

Jailbreak Unchecked File Operation Result Code\Path 65:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=65
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/common/Helper.m	/source code/foodintaketracker/common/Helper.m
Line	120	120
Object	writeToFile:atomically:	writeToFile:atomically:

Code Snippet	
File Name	/source code/foodintaketracker/common/Helper.m
Method	+ (NSString *)saveVoiceRecording:(NSData *)data {

```
....
120.       [data writeToFile:filePath atomically:YES];
```

Unchecked Return Value

Description

Unchecked Return Value\Path 23:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=23
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/CArray.m	/source code/library/bnpiechart/CArray.m
Line	23	23
Object	malloc	malloc

Code Snippet	
File Name	/source code/library/bnpiechart/CArray.m
Method	CArray *CArrayNew(int capacity, size_t elementSize) {

```
....
23.     CArray *cArray = malloc(sizeof(CArray));
```

Unchecked Return Value\Path 24:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=24
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/datahelper.m	/source code/Helpers/datahelper.m
Line	105	105
Object	sprintf	sprintf

Code Snippet	
File Name	/source code/Helpers/datahelper.m
Method	+ (NSString *)getDeviceIdentifier {

```
....
105.                            sprintf(macaddrstr,
"%02x:%02x:%02x:%02x:%02x:%02x",
```

Unchecked Return Value\Path 25:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=25
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/CArray.m	/source code/library/bnpiechart/CArray.m
Line	15	15
Object	elements	elements

Code Snippet	
File Name	/source code/library/bnpiechart/CArray.m
Method	CArray *CArrayInit(CArray *cArray, int capacity, size_t elementSize) {

```
....
15.       cArray->elements = malloc(elementSize * capacity);
```

Buffer Size Literal Condition

[Description]()

Buffer Size Literal Condition\Path 127:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=127
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/BNColor.m	/source code/library/bnpiechart/BNColor.m
Line	42	43
Object	3	3

Code Snippet

File Name /source code/library/bnpiechart/BNColor.m

Method + (BNColor *)colorFromRGBHexString:(NSString *)rgbString {

```
....
42.     int component[3];
43.     for (int i = 0; i < 3; ++i) {
```

Buffer Size Literal Condition\Path 128:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=128
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/bnpiechart.m	/source code/library/bnpiechart/bnpiechart.m
Line	239	250
Object	4	4

Code Snippet

File Name /source code/library/bnpiechart/bnpiechart.m

Method - (void)addLabelForLastName {

```
....
239.        float cornerAngles[4];
....
250.        for (i = 0; i < 4 && rayAngle > cornerAngles[i]; ++i);
```

Buffer Size Literal Condition\Path 129:

Severity	Low

Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=129
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/bnpiechart.m	/source code/library/bnpiechart/bnpiechart.m
Line	300	311
Object	4	4

Code Snippet
File Name /source code/library/bnpiechart/bnpiechart.m
Method - (void)addImages {

```
....
300.      float cornerAngles[4];
....
311.      for (i = 0; i < 4 && rayAngle > cornerAngles[i]; ++i);
```

Log Forging

Description

Log Forging\Path 1:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=1
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/addfoodviewcontroller.m	/source code/Helpers/logginghelper.m
Line	200	44
Object	text	NSLog

Code Snippet
File Name /source code/foodintaketracker/controller/addfoodviewcontroller.m
Method - (void)textFieldDidChange:(NSNotification *)notification {

```
....
200.          NSString *searchText = textField.text;
```

File Name /source code/Helpers/logginghelper.m
Method +(void)logMethodEntrance:(NSString *)methodName paramNames:(NSArray *)paramNames params:(NSArray *)params {

```
....
44.            NSLog(@"%@]]", log);
```

Log Forging\Path 2:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=2
Result Comment	
Status	New

	Source	Destination
File	/source code/foodintaketracker/controller/addfoodviewcontroller.m	/source code/Helpers/logginghelper.m
Line	200	53
Object	text	NSLog

Code Snippet
File Name /source code/foodintaketracker/controller/addfoodviewcontroller.m
Method - (void)textFieldDidChange:(NSNotification *)notification {

```
....
200.         NSString *searchText = textField.text;
```

File Name /source code/Helpers/logginghelper.m
Method +(void)logMethodExit:(NSString *)methodName returnValue:(id)value {

```
....
53.            NSLog(@"[Output parameter %@]", value);
```

Incorrect Initialization

Description

Incorrect Initialization\Path 15:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=15
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/nsobject+be.m	/source code/library/bnpiechart/nsobject+be.m
Line	24	24
Object	alloc	alloc

Code Snippet	
File Name	/source code/library/bnpiechart/nsobject+be.m
Method	+ (BeProxy *)beProxyForClass:(Class)class {

```
....
24.     BeProxy *beProxy = [BeProxy alloc];
```

Incorrect Initialization\Path 16:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=16
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/nsobject+be.m	/source code/library/bnpiechart/nsobject+be.m
Line	25	25
Object	alloc	alloc

Code Snippet	
File Name	/source code/library/bnpiechart/nsobject+be.m
Method	+ (BeProxy *)beProxyForClass:(Class)class {

```
....
25.     beProxy->target = [class alloc];
```

Functions Apple Recommends To Avoid

Description

Functions Apple Recommends To Avoid\Path 14:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=14
Result Comment	
Status	New

	Source	Destination
File	/source code/Helpers/datahelper.m	/source code/Helpers/datahelper.m
Line	105	105
Object	sprintf	sprintf

Code Snippet	
File Name	/source code/Helpers/datahelper.m
Method	+ (NSString *)getDeviceIdentifier {

```
....
105.                            sprintf(macaddrstr,
"%02x:%02x:%02x:%02x:%02x:%02x",
```

Use of Insufficiently Random Values

Description

Use of Insufficiently Random Values\Path 26:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=26
Result Comment	
Status	New

	Source	Destination
File	/source code/library/bnpiechart/BNColor.m	/source code/library/bnpiechart/BNColor.m
Line	201	201
Object	rand	rand

Code Snippet

File Name	/source code/library/bnpiechart/BNColor.m
Method	+ (BNColor *)randomBrightColor {

```
....
201.     float hue = (float)rand() / RAND_MAX;
```

Unchecked CString Convertion

Description

Unchecked CString Convertion\Path 66:

Severity	Low
Result State	To Verify
Online Results	https://cxcloudscan.com/CxWebClient/ViewerMain.aspx?scanid=11440&projectid=441&pathid=66
Result Comment	
Status	New

	Source	Destination
File	/source code/library/postgresql/pgclientkit/pgclientparams.m	/source code/library/postgresql/pgclientkit/pgclientparams.m
Line	76	76
Object	cStringUsingEncoding:	cStringUsingEncoding:

Code Snippet

File Name	/source code/library/postgresql/pgclientkit/pgclientparams.m
Method	void _paramSetText(PGClientParams* params,NSUInteger i,NSString* text,NSStringEncoding encoding,Oid pgtype) {

```
....
76.   params->values[i] = [text cStringUsingEncoding:encoding];
```

Third Party Keyboards On Sensitive Field

Risk

What might happen

A malicious third party keyboard may perform key-logging and transmit the logged information to an attacker’s server. By default, all UITextField and UITextView objects with the secureTextEntry property set to YES disallow the usage of third party keyboards. However, if secureTextEntry is set to NO, to make the characters visible for example, a third party keyboard may steal sensitive data.

Cause

How does it happen

The user installs a malicious third party keyboard that presents itself as legitimate. The user taps a UITextField in the application UI to enter his social security number. The UITextField has its secureTextEntry property set to NO, so that typed-in characters are visible. In this case, third party keyboards are available for use. The user chooses to use the malicious keyboard. The keyboard logs the entered social security number and sends it to an attacker's server.

General Recommendations

How to avoid it

1.	Mark the UITextView and UITextField objects that contain sensitive data as secure by setting the secureTextEntry property to YES. See example “Marking a UITextField as secure”.
-Or-
2.	Disable the usage of third party keyboards entirely in your application. Note that this may degrade the user experience of your application. See example “To disabling third party keyboards, add the following code to your UIApplicationDelegate”.

Source Code Examples

Objective-C
Marking a UITextField as secure

```
UITextField* textField = [[UITextFieldalloc] init];
textField.secureTextEntry = YES;
```

To disabling third party keyboards, add the following code to your UIApplicationDelegate

```
-(BOOL)application:(UIApplication *)application
shouldAllowExtensionPointIdentifier:(NSString *)extensionPointIdentifier
{

    if (extensionPointIdentifier == UIApplicationKeyboardExtensionPointIdentifier)
    {
        return NO;
    }

    return YES;
}
```

Cut And Paste Leakage

Risk

What might happen

An attacker could get access to the data stored in the cut and paste buffer. If sensitive data was stored in the buffer, it could be leaked.

Cause

How does it happen

The application shows some sensitive data on a screen, such as a credit card number, using a UI element that allows copying its contents to the paste buffer. The user finishes using the application, and closes it. The sensitive data remains in the paste buffer. An attacker steals the mobile device, and pastes this data into some other application, such as email.

General Recommendations

How to avoid it

1. Mark UITextView and UITextField that contain sensitive data as secure by setting the secureTextEntry property to YES. -Or-
2. Disable the “copy” action on UITextView and UITextField instances that contain sensitive data. -Or-
3. Clear the paste board buffer just before the application moves to the background. Listen to UIApplicationDidEnterBackgroundNotification or implement an applicationDidEnterBackground: method in UIApplicationDelegate, and clear the paste board using this method.

Source Code Examples

Objective-C

To disable the copy action on UITextField, implement a new class inheriting from UITextFiled, and implement canPerformAction:withSender:

```
/* CXTextField.h file */

#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>
@interface CXTextField : UITextField
@end

/* CXTextField.m file */

#import "CXTextField.h"
@implementation CXTextField

- (BOOL)canPerformAction:(SEL)action withSender:(id)sender
{
if (sender == [UIApplicationsharedApplication] && action == @selector(copy:))
    {
return NO;
    } else {
return [super canPerformAction:actionwithSender:sender];
    }
}

@end
```

The example below shows how to clear the paste buffer just before the application goes to background:

```
- (void)viewDidLoad
{

    [[NSNotificationCenterdefaultCenter] addObserver:self
                                selector:@selector(didEnterBackground:)
                                name:UIApplicationDidEnterBackgroundNotification
                        object:nil];
}

-(void)didEnterBackground:(NSNotification *)notification
{
UIPasteboard *pasteboard = [UIPasteboardgeneralPasteboard];
pasteboard.string = @"";
}
```

Marking a UITextField as secure:

```
UITextField* textField = [[UITextFieldalloc] init];
textField.secureTextEntry = YES;
```

Insecure Storage of Sensitive Information

Weakness ID: 922 *(Weakness Class)* **Status:** Incomplete

Description

Description Summary

The software stores sensitive information without properly limiting read or write access by unauthorized actors.

Extended Description

If read access is not properly restricted, then attackers can steal the sensitive information. If write access is not properly restricted, then attackers can modify and possibly delete the data, causing incorrect results and possibly a denial of service.

Time of Introduction

- Architecture and Design
- Implementation
- System Configuration

Applicable Platforms

Languages

Language-independent

Common Consequences

Scope	Effect
Confidentiality	Technical Impact: *Read application data; Read files or directories* Attackers can read sensitive information by accessing the unrestricted storage mechanism.
Integrity	Technical Impact: *Modify application data; Modify files or directories* Attackers can read sensitive information by accessing the unrestricted storage mechanism.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	664		**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	312		Development Concepts699 Research Concepts1000
ParentOf	Weakness Base	921		**Development Concepts (primary)699** **Research Concepts (primary)1000**

Relationship Notes

There is an overlapping relationship between insecure storage of sensitive information () and missing encryption of sensitive information (). Encryption is often used to prevent an attacker from reading the sensitive data. However, encryption does not prevent the attacker from erasing or overwriting the data.

Maintenance Notes

This is a high-level node that includes children from various parts of the CWE research view (). Currently, most of the information is in these child entries. This entry will be made more comprehensive in later CWE versions.

Content History

Submissions

Submission Date	Submitter	Organization	Source
2013-06-23	MITRE		Internal CWE Team

Privacy Violation

Weakness ID: 359 *(Weakness Class)* **Status:** Incomplete

Description

Description Summary

Mishandling private information, such as customer passwords or social security numbers, can compromise user privacy and is often illegal.

Time of Introduction

- Architecture and Design
- Implementation
- Operation

Applicable Platforms

Languages

All

Demonstrative Examples

Example 1

The following code contains a logging statement that tracks the contents of records added to a database by storing them in a log file. Among other values that are stored, the getPassword() function returns the user-supplied plaintext password associated with the account.

(Bad Code)

Example Language: **C#**

```
pass = GetPassword();
...
dbmsLog.WriteLine(id + ":" + pass + ":" + type + ":" + tstamp);
```

The code in the example above logs a plaintext password to the filesystem. Although many developers trust the filesystem as a safe storage location for data, it should not be trusted implicitly, particularly when privacy is a concern.

Other Notes

Privacy violations occur when: 1. Private user information enters the program. 2. The data is written to an external location, such as the console, file system, or network.

Private data can enter a program in a variety of ways:

- Directly from the user in the form of a password or personal information
- Accessed from a database or other data store by the application
- Indirectly from a partner or other third party

Sometimes data that is not labeled as private can have a privacy implication in a different context. For example, student identification numbers are usually not considered private because there is no explicit and publicly-available mapping to an individual student's personal information. However, if a school generates identification numbers based on student social security numbers, then the identification numbers should be considered private.

Security and privacy concerns often seem to compete with each other. From a security perspective, you should record all important operations so that any anomalous activity can later be identified. However, when private data is involved, this practice can in fact create risk. Although there are many ways in which private data can be handled unsafely, a common risk stems from misplaced trust. Programmers often trust the operating environment in which a program runs, and therefore believe that it is acceptable store private information on the file system, in the registry, or in other locally-controlled resources. However, even if access to certain resources is restricted, this does not guarantee that the individuals who do have access can be trusted.

For example, in 2004, an unscrupulous employee at AOL sold approximately 92 million private customer e-mail addresses to a spammer marketing an offshore gambling web site. In response to such high-profile exploits, the collection and management of private data is becoming increasingly regulated. Depending on its location, the type of business it conducts, and the nature of any private data it handles, an organization may be required to comply with one or more of the following federal and state regulations: - Safe Harbor Privacy Framework [REF-2] - Gramm-Leach Bliley Act (GLBA) [REF-3] - Health Insurance Portability and Accountability Act (HIPAA) [REF-4] - California SB-1386 [REF-5]

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	200	Information Exposure	**Research Concepts (primary)1000**

ChildOf	Category	254	Security Features	**Development Concepts (primary)699** **Seven Pernicious Kingdoms (primary)700**
ParentOf	Weakness Variant	202	Privacy Leak through Data Queries	**Research Concepts (primary)1000**

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Privacy Violation

References

J. Oates. "AOL man pleads guilty to selling 92m email addies". The Register. 2005. <http://www.theregister.co.uk/2005/02/07/aol_email_theft/>.

[REF-2] U.S. Department of Commerce. "Safe Harbor Privacy Framework". <http://www.export.gov/safeharbor/>.

[REF-3] Federal Trade Commission. "Financial Privacy: The Gramm-Leach Bliley Act (GLBA)". <http://www.ftc.gov/privacy/glbact/index.html>.

[REF-4] U.S. Department of Human Services. "Health Insurance Portability and Accountability Act (HIPAA)". <http://www.hhs.gov/ocr/hipaa/>.

[REF-5] Government of the State of California. "California SB-1386". 2002. <http://info.sen.ca.gov/pub/01-02/bill/sen/sb_1351-1400/sb_1386_bill_20020926_chaptered.html>.

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	7 Pernicious Kingdoms		Externally Mined
Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-07-01	Eric Dalci updated Time of Introduction	Cigital	External
2008-09-08	CWE Content Team updated Relationships, Other Notes, Taxonomy Mappings	MITRE	Internal
2009-03-10	CWE Content Team updated Other Notes	MITRE	Internal
2009-07-27	CWE Content Team updated Demonstrative Examples	MITRE	Internal
2009-12-28	CWE Content Team updated Other Notes, References	MITRE	Internal
2010-02-16	CWE Content Team updated Other Notes, References	MITRE	Internal

BACK TO TOP

Uncontrolled Format String

Weakness ID: 134 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The software uses externally-controlled format strings in printf-style functions, which can lead to buffer overflows or data representation problems.

Time of Introduction

- Implementation

Applicable Platforms

Languages

C: *(Often)*

C++: *(Often)*

Perl: *(Rarely)*

Languages that support format strings

Modes of Introduction

The programmer rarely intends for a format string to be user-controlled at all. This weakness is frequently introduced in code that constructs log messsages, where a constant format string is omitted.

In cases such as localization and internationalization, the language-specific message repositories could be an avenue for exploitation, but the format string issue would be resultant, since attacker control of those repositories would also allow modification of message length, format, and content.

Common Consequences

Scope	Effect
Confidentiality	Format string problems allow for information disclosure which can severely simplify exploitation of the program.
Access Control	Format string problems can result in the execution of arbitrary code.

Likelihood of Exploit

Very High

Detection Methods

Automated Static Analysis

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

Black Box

Since format strings often occur in rarely-occurring erroneous conditions (e.g. for error message logging), they can be difficult to detect using black box methods. It is highly likely that many latent issues exist in executables that do not have associated source code (or equivalent source.

Effectiveness: Limited

Demonstrative Examples

Example 1

The following example is exploitable, due to the printf() call in the printWrapper() function. Note: The stack buffer was added to make exploitation more simple.

(Bad Code)

Example Language: **C**

```
#include <stdio.h>

void printWrapper(char *string) {

printf(string);
}
```

```
int main(int argc, char **argv) {

char buf[5012];
memcpy(buf, argv[1], 5012);
printWrapper(argv[1]);
return (0);
}
```

Example 2

The following code copies a command line argument into a buffer using snprintf().

(Bad Code)

Example Language: **C**

```
int main(int argc, char **argv){
char buf[128];
...
snprintf(buf,128,argv[1]);
}
```

This code allows an attacker to view the contents of the stack and write to the stack using a command line argument containing a sequence of formatting directives. The attacker can read from the stack by providing more formatting directives, such as %x, than the function takes as arguments to be formatted. (In this example, the function takes no arguments to be formatted.) By using the %n formatting directive, the attacker can write to the stack, causing snprintf() to write the number of bytes output thus far to the specified argument (rather than reading a value from the argument, which is the intended behavior). A sophisticated version of this attack will use four staggered writes to completely control the value of a pointer on the stack.

Example 3

Certain implementations make more advanced attacks even easier by providing format directives that control the location in memory to read from or write to. An example of these directives is shown in the following code, written for glibc:

(Bad Code)

Example Language: **C**

```
printf("%d %d %1$d %1$d\n", 5, 9);
```

This code produces the following output: 5 9 5 5 It is also possible to use half-writes (%hn) to accurately control arbitrary DWORDS in memory, which greatly reduces the complexity needed to execute an attack that would otherwise require four staggered writes, such as the one mentioned in the first example.

Observed Examples

Reference	Description
CVE-2002-1825	format string in Perl program
CVE-2001-0717	format string in bad call to syslog function
CVE-2002-0573	format string in bad call to syslog function
CVE-2002-1788	format strings in NNTP server responses
CVE-2007-2027	Chain: untrusted search path enabling resultant format string by loading malicious internationalization messages

Potential Mitigations

Phase: Requirements

Choose a language that is not subject to this flaw.

Phase: Implementation

Ensure that all format string functions are passed a static string which cannot be controlled by the user and that the proper number of arguments are always sent to that function as well. If at all possible, use functions that do not support the %n operator in format strings.

Build: Heed the warnings of compilers and linkers, since they may alert you to improper usage.

Other Notes

While Format String vulnerabilities typically fall under the Buffer Overflow category, technically they are not overflowed buffers. The Format String vulnerability is fairly new (circa 1999) and stems from the fact that there is no realistic way for a function that takes a variable number of arguments to determine just how many arguments were passed in. The most common functions that take a variable number of arguments, including C-runtime functions, are the printf() family of calls. The Format String problem appears in a number of ways. A *printf() call without a format specifier is dangerous and can be exploited. For example, printf(input); is exploitable, while printf(y, input); is not exploitable in that context. The result of the first call, used incorrectly, allows for an attacker to be able to peek at stack memory since the input string will be used as the format specifier. The attacker can stuff the input string with format specifiers and begin reading stack values, since the remaining parameters will be pulled from the stack. Worst case, this improper use may give away enough control to allow an arbitrary value (or values in the case of an exploit program) to be written into the memory of the running program.

Frequently targeted entities are file names, process names, identifiers.

Format string problems are a classic C/C++ issue that are now rare due to the ease of discovery. One main reason format string vulnerabilities can be exploited is due to the %n operator. The %n operator will write the number of characters, which have been printed by the format string therefore far, to the memory pointed to by its argument. Through skilled creation of a format string, a malicious user may use values on the stack to create a write-what-where condition. Once this is achieved, he can execute arbitrary code. Other operators can be used as well; for example, a %9999s operator could also trigger a buffer overflow, or when used in file-formatting functions like fprintf, it can generate a much larger output than intended.

Weakness Ordinalities

Ordinality	Description
Primary	*(where the weakness exists independent of other weaknesses)*

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	20	Improper Input Validation	**Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Class	74	Failure to Sanitize Data into a Different Plane ('Injection')	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ChildOf	Category	133	String Errors	Development Concepts699
ChildOf	Category	633	Weaknesses that Affect Memory	**Resource-specific Weaknesses (primary)631**
ChildOf	Category	726	OWASP Top Ten 2004 Category A5 - Buffer Overflows	**Weaknesses in OWASP Top Ten (2004) (primary)711**
ChildOf	Category	743	CERT C Secure Coding Section 09 - Input Output (FIO)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
ChildOf	Category	808	2010 Top 25 - Weaknesses On the Cusp	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**
PeerOf	Weakness Base	123	Write-what-where Condition	Research Concepts1000
MemberOf	View	630	Weaknesses Examined by SAMATE	**Weaknesses Examined by SAMATE (primary)630**
MemberOf	View	635	Weaknesses Used by NVD	**Weaknesses Used by NVD (primary)635**

Research Gaps

Format string issues are under-studied for languages other than C. Memory or disk consumption, control flow or variable alteration, and data corruption may result from format string exploitation in applications written in other languages such as Perl, PHP, Python, etc.

Affected Resources

- Memory

Functional Areas

- logging
- errors
- general output

f Causal Nature

Implicit

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Format string vulnerability

7 Pernicious Kingdoms			Format String
CLASP			Format string problem
CERT C Secure Coding	FIO30-C	Exact	Exclude user input from format strings
OWASP Top Ten 2004	A1	CWE More Specific	Unvalidated Input
CERT C Secure Coding	FIO30-C		Exclude user input from format strings
WASC	6		Format String

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
67	String Format Overflow in syslog()	

White Box Definitions

A weakness where the code path has:

1. start statement that accepts input

2. end statement that passes a format string to format string function where

a. the input data is part of the format string and

b. the format string is undesirable

Where "undesirable" is defined through the following scenarios:

1. not validated

2. incorrectly validated

References

Steve Christey. "Format String Vulnerabilities in Perl Programs". <http://www.securityfocus.com/archive/1/418460/30/0/threaded>.

Hal Burch and Robert C. Seacord. "Programming Language Format String Vulnerabilities". <http://www.ddj.com/dept/security/197002914>.

Tim Newsham. "Format String Attacks". Guardent. September 2000. <http://www.lava.net/~newsham/format-string-attacks.pdf>.

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 5, "Format String Bugs" Page 147. 2nd Edition. Microsoft. 2002.

Content History

Submissions

Submission Date	Submitter	Organization	Source
	PLOVER		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-08-01	added/updated white box definitions	KDM Analytics	External
2008-09-08	CWE Content Team updated Applicable Platforms, Common Consequences, Detection Factors, Modes of Introduction, Relationships, Other Notes, Research Gaps, Taxonomy Mappings, Weakness Ordinalities	MITRE	Internal
2008-11-24	CWE Content Team updated Relationships, Taxonomy Mappings	MITRE	Internal
2009-03-10	CWE Content Team updated Relationships	MITRE	Internal
2009-05-27	CWE Content Team updated Demonstrative Examples	MITRE	Internal
2009-07-17	KDM Analytics Improved the White Box Definition		External
2009-07-27	CWE Content Team updated White Box Definitions	MITRE	Internal
2010-02-16	CWE Content Team updated Detection Factors, References, Relationships, Taxonomy Mappings	MITRE	Internal

BACK TO TOP

Screen Caching

Risk

What might happen

An attacker could get access to the application screenshots that were saved by the system. If sensitive data was presented in the screenshot, it could be leaked.

Cause

How does it happen

The application shows some sensitive data on a screen, such as a credit card numbers, using a regular UI element. The user presses the home button and sends the active application to the background. At the same moment, the system takes a screenshot of the application screen and saves the screenshot in a system folder. The screenshot is later used in an App Switcher to present open application screen previews. An attacker steals the mobile device, gets access to the screenshots folder, and steals the sensitive information.

General Recommendations

How to avoid it

1. Mark UITextView and UITextField that contain sensitive data as secure by setting the secureTextEntry property to YES. -Or-
2. Make the sensitive UI elements invisible just before the application moves to the background, and the screenshot is taken. To accomplish this, listen to UIApplicationDidEnterBackgroundNotification or implement an applicationDidEnterBackground: method in UIApplicationDelegate, and hide the sensitive UI element in this method.

Source Code Examples

Objective-C

Marking a UITextField as secure:

```
UITextField* textField = [[UITextFieldalloc] init];
textField.secureTextEntry = YES;
```

The example below shows how to hide a sensitive UI element just before the application goes to background:

```
- (void)viewDidLoad
{

    [[NSNotificationCenterdefaultCenter] addObserver:self
                                 selector:@selector(didEnterBackground:)
                         name:UIApplicationDidEnterBackgroundNotification
                         object:nil];

    [[NSNotificationCenterdefaultCenter] addObserver:self
selector:@selector(didBecomeActive:)
name:UIApplicationDidBecomeActiveNotification
object:nil];

}

-(void)didEnterBackground:(NSNotification *)notification
```

```
{
    [self.creditCardNumberTextFieldsetHidden:YES];
}

-(void)didBecomeActive:(NSNotification *)notification
{
    [self.creditCardNumberTextFieldsetHidden:NO];
}
```

External Control of File Name or Path

Weakness ID: 73 *(Weakness Class)* **Status:** Draft

Description

Description Summary

The software allows user input to control or influence paths or file names that are used in filesystem operations.

Extended Description

This could allow an attacker to access or modify system files or other files that are critical to the application.

Path manipulation errors occur when the following two conditions are met:

1. An attacker can specify a path used in an operation on the filesystem.

2. By specifying the resource, the attacker gains a capability that would not otherwise be permitted.

For example, the program may give the attacker the ability to overwrite the specified file or run with a configuration controlled by the attacker.

Time of Introduction

- Architecture and Design
- Implementation
- Operation

Applicable Platforms

Languages

All

Operating Systems

UNIX: *(Often)*

Windows: *(Often)*

Mac OS: *(Often)*

Common Consequences

Scope	Effect
Confidentiality	The application can operate on unexpected files. Confidentiality is violated when the targeted filename is not directly readable by the attacker.
Integrity	The application can operate on unexpected files. This may violate integrity if the filename is written to, or if the filename is for a program or other form of executable code.
Availability	The application can operate on unexpected files. Availability can be violated if the attacker specifies an unexpected file that the application modifies. Availability can also be affected if the attacker specifies a filename for a large file, or points to a special device or a file that does not have the format that the application expects.

Likelihood of Exploit

High to Very High

Detection Methods

Automated Static Analysis

The external control or influence of filenames can often be detected using automated static analysis that models data flow within the software.

Automated static analysis might not be able to recognize when proper input validation is being performed, leading to false positives - i.e., warnings that do not have any security consequences or require any code changes.

Demonstrative Examples

Example 1

The following code uses input from an HTTP request to create a file name. The programmer has not considered the possibility that an attacker could provide a file name such as "../../tomcat/conf/server.xml", which causes the application to delete one of its own configuration files (CWE-22).

(Bad Code)

Example Language: **Java**

```
String rName = request.getParameter("reportName");
File rFile = new File("/usr/local/apfr/reports/" + rName);
...
rFile.delete();
```

Example 2

The following code uses input from a configuration file to determine which file to open and echo back to the user. If the program runs with privileges and malicious users can change the configuration file, they can use the program to read any file on the system that ends with the extension .txt.

(Bad Code)

Example Language: **Java**

```
fis = new FileInputStream(cfg.getProperty("sub")+".txt");
amt = fis.read(arr);
out.println(arr);
```

Observed Examples

Reference	Description
CVE-2008-5748	Chain: external control of values for user's desired language and theme enables path traversal.
CVE-2008-5764	Chain: external control of user's target language enables remote file inclusion.

Potential Mitigations

Phase: Architecture and Design

When the set of filenames is limited or known, create a mapping from a set of fixed input values (such as numeric IDs) to the actual filenames, and reject all other inputs. For example, ID 1 could map to "inbox.txt" and ID 2 could map to "profile.txt". Features such as the ESAPI AccessReferenceMap provide this capability.

Phases: Architecture and Design; Operation

Run your code in a "jail" or similar sandbox environment that enforces strict boundaries between the process and the operating system. This may effectively restrict all access to files within a particular directory.

Examples include the Unix chroot jail and AppArmor. In general, managed code may provide some protection.

This may not be a feasible solution, and it only limits the impact to the operating system; the rest of your application may still be subject to compromise.

Be careful to avoid CWE-243 and other weaknesses related to jails.

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.

Phase: Implementation

Strategy: Input Validation

Assume all input is malicious. Use an "accept known good" input validation strategy, i.e., use a whitelist of acceptable inputs that strictly conform to specifications. Reject any input that does not strictly conform to specifications, or transform it into something that does. Do not rely exclusively on looking for malicious or malformed inputs (i.e., do not rely on a blacklist). However, blacklists can be useful for detecting potential attacks or determining which inputs are so malformed that they should be rejected outright.

When performing input validation, consider all potentially relevant properties, including length, type of input, the full range of acceptable values, missing or extra inputs, syntax, consistency across related fields, and conformance to business rules. As an example of business rule logic, "boat" may be syntactically valid because it only contains alphanumeric characters, but it is not valid if you are expecting colors such as "red" or "blue."

For filenames, use stringent whitelists that limit the character set to be used. If feasible, only allow a single "." character in the filename to avoid weaknesses such as CWE-23, and exclude directory separators such as "/" to avoid CWE-36. Use a whitelist of allowable file extensions, which will help to avoid CWE-434.

Phase: Implementation

Use a built-in path canonicalization function (such as realpath() in C) that produces the canonical version of the pathname, which effectively removes ".." sequences and symbolic links (CWE-23, CWE-59).

Phases: Installation; Operation

Use OS-level permissions and run as a low-privileged user to limit the scope of any successful attack.

Phases: Operation; Implementation

If you are using PHP, configure your application so that it does not use register_globals. During implementation, develop your application so that it does not rely on this feature, but be wary of implementing a register_globals emulation that is subject to weaknesses such as CWE-95, CWE-621, and similar issues.

Phase: Testing

Use automated static analysis tools that target this type of weakness. Many modern techniques use data flow analysis to minimize the number of false positives. This is not a perfect solution, since 100% accuracy and coverage are not feasible.

Phase: Testing

Use 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.

Phase: Testing

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

Weakness Ordinalities

Ordinality	Description
Primary	*(where the weakness exists independent of other weaknesses)*

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Class	20	Improper Input Validation	**Development Concepts (primary)699** **Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Class	610	Externally Controlled Reference to a Resource in Another Sphere	Research Concepts1000
ChildOf	Weakness Class	642	External Control of Critical State Data	**Research Concepts (primary)1000**
ChildOf	Category	723	OWASP Top Ten 2004 Category A2 - Broken Access Control	**Weaknesses in OWASP Top Ten (2004) (primary)711**
ChildOf	Category	752	2009 Top 25 - Risky Resource Management	**Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)750**
CanPrecede	Weakness Class	22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	Research Concepts1000
CanPrecede	Weakness Base	41	Improper Resolution of Path Equivalence	Research Concepts1000
CanPrecede	Weakness Base	59	Improper Link Resolution Before File Access ('Link Following')	Research Concepts1000
CanPrecede	Weakness Base	98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion')	Research Concepts1000
CanPrecede	Weakness Base	434	Unrestricted Upload of File with Dangerous Type	Research Concepts1000
CanAlsoBe	Weakness Base	99	Improper Control of Resource Identifiers ('Resource Injection')	Research Concepts1000

Relationship Notes

The external control of filenames can be the primary link in chains with other file-related weaknesses, as seen in the CanPrecede relationships. This is because software systems use files for many different purposes: to execute programs, load code libraries, to store application data, to store configuration settings, record temporary data, act as signals or semaphores to other processes, etc.

However, those weaknesses do not always require external control. For example, link-following weaknesses (CWE-59) often involve pathnames that are not controllable by the attacker at all.

The external control can be resultant from other issues. For example, in PHP applications, the register_globals setting can allow an attacker to modify variables that the programmer thought were immutable, enabling file inclusion (CWE-98) and path traversal (CWE-22). Operating with excessive privileges (CWE-250) might allow an attacker to specify an input filename that is not directly readable by the attacker, but is accessible to the privileged program. A buffer overflow (CWE-119) might give an attacker control over nearby memory locations that are related to pathnames, but were not directly modifiable by the attacker.

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Path Manipulation

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
13	Subverting Environment Variable Values	
64	Using Slashes and URL Encoding Combined to Bypass Validation Logic	
72	URL Encoding	
78	Using Escaped Slashes in Alternate Encoding	
79	Using Slashes in Alternate Encoding	
76	Manipulating Input to File System Calls	
80	Using UTF-8 Encoding to Bypass Validation Logic	

References

"OWASP Enterprise Security API (ESAPI) Project". <http://www.owasp.org/index.php/ESAPI>.

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	7 Pernicious Kingdoms		Externally Mined

Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-07-01	Eric Dalci updated Time of Introduction	Cigital	External
2008-09-08	CWE Content Team updated Relationships, Other Notes, Taxonomy Mappings, Weakness Ordinalities	MITRE	Internal
2009-01-12	CWE Content Team updated Applicable Platforms, Causal Nature, Common Consequences, Demonstrative Examples, Description, Observed Examples, Other Notes, Potential Mitigations, References, Relationship Notes, Relationships, Weakness Ordinalities	MITRE	Internal
2009-03-10	CWE Content Team updated Potential Mitigations, Relationships	MITRE	Internal
2009-07-27	CWE Content Team updated Demonstrative Examples	MITRE	Internal
2009-10-29	CWE Content Team updated Common Consequences, Description	MITRE	Internal
2009-12-28	CWE Content Team updated Detection Factors	MITRE	Internal
2010-02-16	CWE Content Team updated Potential Mitigations	MITRE	Internal

Previous Entry Names	
Change Date	**Previous Entry Name**
2008-04-11	Path Manipulation

BACK TO TOP

Autocorrection Keystroke Logging

Risk

What might happen

An attacker could get access to the store of autocorrection exception strings that were saved by the system. If sensitive data was saved in this store, it could be leaked.

Cause

How does it happen

The application presents a text input UI element to the user for sensitive data, such as a security question. The user types in the answer. The input UI element checks with the autocorrection system if the input needs correction, and presents a suggestion to the user. The user cancels the suggestion. The input string is saved in a system repository of strings that will not be corrected in the future. An attacker steals the mobile device, gets access to the autocorrection store, and steals the answer to the security question.

General Recommendations

How to avoid it

1. Mark UITextView and UITextField that contain sensitive data as secure by setting the secureTextEntry property to YES. -Or-
2. Disable the autocorrection mechanism on sensitive UI elements.

Source Code Examples

Objective-C
Marking a UITextFieldas secure:

```
UITextField* textField = [[UITextFieldalloc] init];
textField.secureTextEntry = YES;
```

Disabling the autocorrection on UITextField input element:

```
UITextField* textField = [[UITextFieldalloc] init];
textField.autocorrectionType = UITextAutocorrectionTypeNo;
```

Log Forging

Risk

What might happen

An attacker could engineer audit logs of security-sensitive actions and lay a false audit trail, potentially implicating an innocent user or hiding an incident.

Cause

How does it happen

The application writes audit logs upon security-sensitive actions. Since the audit log includes user input that is neither checked for data type validity nor subsequently sanitized, the input could contain false information made to look like legitimate audit log data,

General Recommendations

How to avoid it

1. Validate all input, regardless of source. Validation should be based on a whitelist: accept only data fitting a specified structure, rather than reject bad patterns.

   Check for:

   - Data type
   - Size
   - Range
   - Format
   - Expected values

2. Validation is not a replacement for encoding. Fully encode all dynamic data, regardless of source, before embedding it in logs.
3. Use a secure logging mechanism.

Source Code Examples

C#

Ensure you encode any special delimiter characters before writing to a log file.

```
Log.Write( logDetails.Replace(CRLF, @"\CRLF"));
```

Java

Ensure you encode any special delimiter characters before writing to a log file.

```
Log.Write( logDetails.Replace(CRLF, @"\CRLF"));
```

Use of Hard-coded Password

Weakness ID: 259 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The software contains a hard-coded password, which it uses for its own inbound authentication or for outbound communication to external components.

Extended Description

A hard-coded password typically leads to a significant authentication failure that can be difficult for the system administrator to detect. Once detected, it can be difficult to fix, so the administrator may be forced into disabling the product entirely. There are two main variations:

Inbound: the software contains an authentication mechanism that checks for a hard-coded password.

Outbound: the software connects to another system or component, and it contains hard-coded password for connecting to that component.

In the Inbound variant, a default administration account is created, and a simple password is hard-coded into the product and associated with that account. This hard-coded password is the same for each installation of the product, and it usually cannot be changed or disabled by system administrators without manually modifying the program, or otherwise patching the software. If the password is ever discovered or published (a common occurrence on the Internet), then anybody with knowledge of this password can access the product. Finally, since all installations of the software will have the same password, even across different organizations, this enables massive attacks such as worms to take place.

The Outbound variant applies to front-end systems that authenticate with a back-end service. The back-end service may require a fixed password which can be easily discovered. The programmer may simply hard-code those back-end credentials into the front-end software. Any user of that program may be able to extract the password. Client-side systems with hard-coded passwords pose even more of a threat, since the extraction of a password from a binary is usually very simple.

Time of Introduction

- Implementation
- Architecture and Design

Applicable Platforms

Languages

Language-independent

Common Consequences

Scope	Effect
Authentication	If hard-coded passwords are used, it is almost certain that malicious users will gain access through the account in question.

Likelihood of Exploit

Very High

Detection Methods

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.

These may be more effective than strictly automated techniques. This is especially the case with weaknesses that are related to design and business rules.

Demonstrative Examples

Example 1

The following code uses a hard-coded password to connect to a database:

(Bad Code)

Example Language: **Java**

```
...
DriverManager.getConnection(url, "scott", "tiger");
...
```

This is an example of an external hard-coded password on the client-side of a connection. This code will run successfully, but anyone who has access to it will have access to the password. Once the program has shipped, there is no going back from the database user "scott" with a password of "tiger" unless the program is patched. A devious employee with access to this information can use it to break into the system. Even worse, if attackers have access to the bytecode for application, they can use the javap -c command to access the disassembled code, which will contain the values of the passwords used. The result of this operation might look something like the following for the example above:

(Attack)

```
javap -c ConnMngr.class
22: ldc #36; //String jdbc:mysql://ixne.com/rxsql
24: ldc #38; //String scott
26: ldc #17; //String tiger
```

Example 2

The following code is an example of an internal hard-coded password in the back-end:

(Bad Code)

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

```
int VerifyAdmin(char *password) {
if (strcmp(password, "Mew!")) {

printf("Incorrect Password!\n");
return(0)
}
printf("Entering Diagnostic Mode...\n");
return(1);
}
```

(Bad Code)

Example Language: **Java**

```
int VerifyAdmin(String password) {
if (passwd.Equals("Mew!")) {
return(0)
}
//Diagnostic Mode
return(1);
}
```

Every instance of this program can be placed into diagnostic mode with the same password. Even worse is the fact that if this program is distributed as a binary-only distribution, it is very difficult to change that password or disable this "functionality."

Potential Mitigations

Phase: Architecture and Design

For outbound authentication: store passwords outside of the code in a strongly-protected, encrypted configuration file or database that is protected from access by all outsiders, including other local users on the same system. Properly protect the key (CWE-320). If you cannot use encryption to protect the file, then make sure that the permissions are as restrictive as possible.

Phase: Architecture and Design

For inbound authentication: Rather than hard-code a default username and password for first time logins, utilize a "first login" mode that requires the user to enter a unique strong password.

Phase: Architecture and Design

Perform access control checks and limit which entities can access the feature that requires the hard-coded password. For example, a feature might only be enabled through the system console instead of through a network connection.

Phase: Architecture and Design

For inbound authentication: apply strong one-way hashes to your passwords and store those hashes in a configuration file or database with appropriate access control. That way, theft of the file/database still requires the attacker to try to crack the password. When handling an incoming password during authentication, take the hash of the password and compare it to the hash that you have saved.

Use randomly assigned salts for each separate hash that you generate. This increases the amount of computation that an attacker needs to conduct a brute-force attack, possibly limiting the effectiveness of the rainbow table method.

Phase: Architecture and Design

For front-end to back-end connections: Three solutions are possible, although none are complete.

The first suggestion involves the use of generated passwords which are changed automatically and must be entered at given time intervals by a system administrator. These passwords will be held in memory and only be valid for the time intervals.

Next, the passwords used should be limited at the back end to only performing actions valid for the front end, as opposed to having full access.

Finally, the messages sent should be tagged and checksummed with time sensitive values so as to prevent replay style attacks.

Phase: Testing

Use monitoring tools that examine the software's process as it interacts with the operating system and the network. This technique is useful in cases when source code is unavailable, if the software was not developed by you, or if you want to verify that the build phase did not introduce any new weaknesses. Examples include debuggers that directly attach to the running process; system-call tracing utilities such as truss (Solaris) and strace (Linux); system activity monitors such as FileMon, RegMon, Process Monitor, and other Sysinternals utilities (Windows); and sniffers and protocol analyzers that monitor network traffic.

Attach the monitor to the process and perform a login. Using disassembled code, look at the associated instructions and see if any of them appear to be comparing the input to a fixed string or value.

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	254	Security Features	**Seven Pernicious Kingdoms (primary)700**
ChildOf	Weakness Base	344	Use of Invariant Value in Dynamically Changing Context	Research Concepts1000
ChildOf	Weakness Class	671	Lack of Administrator Control over Security	Research Concepts1000
ChildOf	Category	724	OWASP Top Ten 2004 Category A3 - Broken Authentication and Session Management	**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	Weakness Base	798	Use of Hard-coded Credentials	**Development Concepts (primary)699** **Research Concepts (primary)1000**
PeerOf	Weakness Base	257	Storing Passwords in a Recoverable Format	Research Concepts1000
PeerOf	Weakness Base	321	Use of Hard-coded Cryptographic Key	Research Concepts1000
MemberOf	View	630	Weaknesses Examined by SAMATE	**Weaknesses Examined by SAMATE (primary)630**
CanFollow	Weakness Base	656	Reliance on Security through Obscurity	Research Concepts1000

f Causal Nature

Explicit

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Password Management: Hard-Coded Password
CLASP			Use of hard-coded password
OWASP Top Ten 2004	A3	CWE More Specific	Broken Authentication and Session Management

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	(CAPEC Version: 1.5)
188	Reverse Engineering	
189	Software Reverse Engineering	
190	Reverse Engineer an Executable to Expose Assumed Hidden Functionality or Content	
191	Read Sensitive Stings Within an Executable	
192	Protocol Reverse Engineering	
205	Lifting credential(s)/key material embedded in client distributions (thick or thin)	

White Box Definitions

Definition: A weakness where code path has:

1. end statement that passes a data item to a password function
2. value of the data item is a constant

Maintenance Notes

This entry should probably be split into multiple variants: an inbound variant (as seen in the second demonstrative example) and an outbound variant (as seen in the first demonstrative example). These variants are likely to have different consequences, detectability, etc. See extended description.

Content History

Submissions

Submission Date	Submitter	Organization	Source
	7 Pernicious Kingdoms		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci updated Time of Introduction	Cigital	External
2008-08-01	 added/updated white box definitions	KDM Analytics	External
2008-08-15	 Suggested OWASP Top Ten 2004 mapping	Veracode	External
2008-09-08	CWE Content Team updated Common Consequences, Relationships, Other Notes, Taxonomy Mappings, Weakness Ordinalities	MITRE	Internal
2008-10-14	CWE Content Team updated Description, Potential Mitigations	MITRE	Internal
2008-11-13	CWE Content Team Significant description modifications to emphasize different variants.	MITRE	Internal
2008-11-24	CWE Content Team updated Demonstrative Examples, Description, Maintenance Notes, Other Notes, Potential Mitigations	MITRE	Internal
2009-01-12	CWE Content Team updated Demonstrative Examples, Description, Maintenance Notes, Potential Mitigations, Relationships	MITRE	Internal
2009-03-10	CWE Content Team updated Potential Mitigations	MITRE	Internal
2009-07-17	KDM Analytics Improved the White Box Definition		External
2009-07-27	CWE Content Team updated Demonstrative Examples, Related Attack Patterns, White Box Definitions	MITRE	Internal
2010-02-16	CWE Content Team updated Demonstrative Examples, Description, Detection Factors, Name, Potential Mitigations, Relationships	MITRE	Internal
2010-04-05	CWE Content Team updated Applicable Platforms	MITRE	Internal

Previous Entry Names

Change Date	Previous Entry Name
2010-02-16	Hard-Coded Password

BACK TO TOP

Missing Initialization

Weakness ID: 456 *(Weakness Base)* **Status:** Draft

Description

Description Summary

The software does not initialize critical variables, which causes the execution environment to use unexpected values.

Time of Introduction

- Implementation

Applicable Platforms

Languages

Language-independent

Demonstrative Examples

Example 1

Here, an uninitialized field in a Java class is used in a seldom-called method, which would cause a NullPointerException to be thrown.

(Bad Code)

Example Language: **Java**

```
private User user;
public void someMethod() {
// Do something interesting.
...

// Throws NPE if user hasn't been properly initialized.
String username = user.getName();
}
```

Example 2

In the following Java code the BankManager class uses the user variable of the class User to allow authorized users to perform bank manager tasks. The user variable is initialized within the method setUser that retrieves the User from the User database. The user is then authenticated as unauthorized user through the method authenticateUser.

(Bad Code)

Example Language: **Java**

```
public class BankManager {

// user allowed to perform bank manager tasks
private User user = null;
private boolean isUserAuthentic = false;

// constructor for BankManager class
public BankManager() {
...
}

// retrieve user from database of users
public User getUserFromUserDatabase(String username){
...
}

// set user variable using username
public void setUser(String username) {
this.user = getUserFromUserDatabase(username);
}
// authenticate user
public boolean authenticateUser(String username, String password) {
if (username.equals(user.getUsername()) && password.equals(user.getPassword())) {
isUserAuthentic = true;
}
return isUserAuthentic;
}

// methods for performing bank manager tasks
...
}
```

However, if the method setUser is not called before authenticateUser then the user variable will not have been initialized and will result in a NullPointerException. The code should verify that the user variable has been initialized before it is used, as in the following code.

(Good Code)

Example Language: **Java**

```
public class BankManager {

// user allowed to perform bank manager tasks
private User user = null;
private boolean isUserAuthentic = false;

// constructor for BankManager class
public BankManager(String username) {
user = getUserFromUserDatabase(username);
}

// retrieve user from database of users
public User getUserFromUserDatabase(String username) {...}

// authenticate user
public boolean authenticateUser(String username, String password) {

if (user == null) {
System.out.println("Cannot find user " + username);
}
else {
if (password.equals(user.getPassword())) {
isUserAuthentic = true;
}
}
return isUserAuthentic;
}

// methods for performing bank manager tasks
...

}
```

Observed Examples

Reference	Description
CVE-2005-2978	Product uses uninitialized variables for size and index, leading to resultant buffer overflow.
CVE-2005-2109	Internal variable in PHP application is not initialized, allowing external modification.
CVE-2005-2193	Array variable not initialized in PHP application, leading to resultant SQL injection.

Potential Mitigations

Check that critical variables are initialized.

Use a static analysis tool to spot non-initialized variables.

Other Notes

This weakness is a major factor in a number of resultant weaknesses, especially in web applications that allow global variable initialization (such as PHP) with libraries that can be directly requested.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Category	452	Initialization and Cleanup Errors	**Development Concepts (primary)699**
ChildOf	Weakness Base	665	Improper Initialization	**Research Concepts (primary)1000**
ChildOf	Category	808	2010 Top 25 - Weaknesses On the Cusp	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**
CanPrecede	Weakness Base	89	Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection')	Research Concepts1000
CanPrecede	Weakness Base	98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion')	Research Concepts1000
CanPrecede	Weakness Base	120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')	Research Concepts1000
ParentOf	Weakness Variant	457	Use of Uninitialized Variable	**Development Concepts (primary)699** **Research Concepts (primary)1000**
CanAlsoBe	Weakness Base	454	External Initialization of Trusted Variables or Data Stores	Research Concepts1000

Research Gaps

It is highly likely that a large number of resultant weaknesses have missing initialization as a primary factor, but researcher reports generally do not provide this level of detail.

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Missing Initialization

Content History

Submissions

Submission Date	Submitter	Organization	Source
	PLOVER		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Sean Eidemiller added/updated demonstrative examples	Cigital	External
2008-07-01	Eric Dalci updated Potential Mitigations, Time of Introduction	Cigital	External
2008-09-08	CWE Content Team updated Relationships, Other Notes, Taxonomy Mappings	MITRE	Internal
2010-02-16	CWE Content Team updated Relationships	MITRE	Internal
2010-04-05	CWE Content Team updated Applicable Platforms, Demonstrative Examples	MITRE	Internal

BACK TO TOP

Unchecked Return Value

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

Description

Description Summary

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

Extended Description

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

Time of Introduction

- Implementation

Applicable Platforms

Languages

All

Common Consequences

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

Likelihood of Exploit

Low

Demonstrative Examples

Example 1

Consider the following code segment:

(Bad Code)

Example Language: **C**

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

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

Example 2

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

(Bad Code)

Example Language: **C**

```
buf = (char*) malloc(req_size);
strncpy(buf, xfer, req_size);
```

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

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

Example 3

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

(Bad Code)

Example Language: **Java**

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

(Bad Code)

Example Language: **Java**

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

Example 4

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

(Bad Code)

Example Language: **Java**

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

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

(Bad Code)

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

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

Example 5

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

(Bad Code)

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

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

Example 6

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

(Bad Code)

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

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

Example 7

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

Example 8

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

(Bad Code)

Example Language: **C**

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

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

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

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

Observed Examples

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

Potential Mitigations

Phase: Implementation

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

Effectiveness: High

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

Phase: Implementation

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

Phase: Implementation

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

Background Details

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

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to	Named Chain(s) this relationship pertains to
ChildOf	Weakness Class	227	Failure to Fulfill API Contract ('API Abuse')	**Development Concepts (primary)699** **Seven Pernicious Kingdoms (primary)700**	
ChildOf	Category	389	Error Conditions, Return Values, Status Codes	Development Concepts699	
ChildOf	Category	728	OWASP Top Ten 2004 Category A7 - Improper Error Handling	**Weaknesses in OWASP Top Ten (2004) (primary)711**	
ChildOf	Category	742	CERT C Secure Coding Section 08 - Memory Management (MEM)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**	
ChildOf	Weakness Class	754	Improper Check for Unusual or Exceptional Conditions	**Research Concepts (primary)1000**	
CanPrecede	Weakness Base	476	NULL Pointer Dereference	Research Concepts1000	Unchecked Return Value to NULL Pointer

					Dereference690
StartsChain	Compound Element: Chain	690	Unchecked Return Value to NULL Pointer Dereference	Named Chains709	Unchecked Return Value to NULL Pointer Dereference690
PeerOf	Weakness Base	273	Improper Check for Dropped Privileges	Research Concepts1000	

Taxonomy Mappings

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

References

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

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

Content History

Submissions

Submission Date	Submitter	Organization	Source
	7 Pernicious Kingdoms		Externally Mined

Modifications

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

BACK TO TOP

Use of Insufficiently Random Values

Weakness ID: 330 *(Weakness Class)* **Status:** Usable

Description

Description Summary

The software may use insufficiently random numbers or values in a security context that depends on unpredictable numbers.

Extended Description

When software generates predictable values in a context requiring unpredictability, it may be possible for an attacker to guess the next value that will be generated, and use this guess to impersonate another user or access sensitive information.

Time of Introduction

- Architecture and Design
- Implementation

Applicable Platforms

Languages

Language-independent

Common Consequences

Scope	Effect
Confidentiality	When a protection mechanism relies on random values to restrict access to a sensitive resource, such as a session ID or a seed for generating a cryptographic key, then the resource being protected could be accessed by guessing the ID or key.
Confidentiality Availability	If software relies on unique, unguessable IDs to identify a resource, an attacker might be able to guess an ID for a resource that is owned by another user. The attacker could then read the resource, or pre-create a resource with the same ID to prevent the legitimate program from properly sending the resource to the intended user. For example, a product might maintain session information in a file whose name is based on a username. An attacker could pre-create this file for a victim user, then set the permissions so that the application cannot generate the session for the victim, preventing the victim from using the application.
Integrity	When an authorization or authentication mechanism relies on random values to restrict access to restricted functionality, such as a session ID or a seed for generating a cryptographic key, then an attacker may access the restricted functionality by guessing the ID or key.

Likelihood of Exploit

Medium to High

Demonstrative Examples

Example 1

The following code uses a statistical PRNG to create a URL for a receipt that remains active for some period of time after a purchase.

(Bad Code)

Example Language: **Java**

```
String GenerateReceiptURL(String baseUrl) {
Random ranGen = new Random();
ranGen.setSeed((new Date()).getTime());
return(baseUrl + ranGen.nextInt(400000000) + ".html");
}
```

This code uses the Random.nextInt() function to generate "unique" identifiers for the receipt pages it generates. Because Random.nextInt() is a statistical PRNG, it is easy for an attacker to guess the strings it generates. Although the underlying design of the receipt system is also faulty, it would be more secure if it used a random number generator that did not produce predictable receipt identifiers, such as a cryptographic PRNG.

Observed Examples

Reference	Description
CVE-2009-3278	Crypto product uses rand() library function to generate a recovery key, making it easier to conduct brute force attacks.
CVE-2009-3238	Random number generator can repeatedly generate the same value.
CVE-2009-2367	Web application generates predictable session IDs, allowing session hijacking.
CVE-2009-2158	Password recovery utility generates a relatively small number of random passwords, simplifying brute force attacks.
CVE-2009-0255	Cryptographic key created with an insufficiently random seed.
CVE-2009-0255	Cryptographic key created with a seed based on the system time.
CVE-2008-5162	Kernel function does not have a good entropy source just after boot.
CVE-2008-4905	Blogging software uses a hard-coded salt when calculating a password hash.
CVE-2008-4929	Bulletin board application uses insufficiently random names for uploaded files, allowing other users to access private files.
CVE-2008-3612	Handheld device uses predictable TCP sequence numbers, allowing spoofing or hijacking of TCP connections.
CVE-2008-2433	Web management console generates session IDs based on the login time, making it easier to conduct session hijacking.
CVE-2008-0166	SSL library uses a weak random number generator that only generates 65,536 unique keys.
CVE-2008-2108	Chain: insufficient precision causes extra zero bits to be assigned, reducing entropy for an API function that generates random numbers.
CVE-2008-2020	CAPTCHA implementation does not produce enough different images, allowing bypass using a database of all possible checksums.
CVE-2008-0087	DNS client uses predictable DNS transaction IDs, allowing DNS spoofing.
CVE-2008-0141	Application generates passwords that are based on the time of day.

Potential Mitigations

Phase: Architecture and Design

Use a well-vetted algorithm that is currently considered to be strong by experts in the field, and select well-tested implementations with adequate length seeds.

In general, if a pseudo-random number generator is not advertised as being cryptographically secure, then it is probably a statistical PRNG and should not be used in security-sensitive contexts.

Pseudo-random number generators can produce predictable numbers if the generator is known and the seed can be guessed. A 256-bit seed is a good starting point for producing a "random enough" number.

Phase: Implementation

Consider a PRNG that re-seeds itself as needed from high quality pseudo-random output sources, such as hardware devices.

Phase: Testing

Use automated static analysis tools that target this type of weakness. Many modern techniques use data flow analysis to minimize the number of false positives. This is not a perfect solution, since 100% accuracy and coverage are not feasible.

Phase: Testing

Perform FIPS 140-2 tests on data to catch obvious entropy problems.

Phase: Testing

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

Phase: Testing

Use monitoring tools that examine the software's process as it interacts with the operating system and the network. This technique is useful in cases when source code is unavailable, if the software was not developed by you, or if you want to verify that the build phase did not introduce any new weaknesses. Examples include debuggers that directly attach to the running process; system-call tracing utilities such as truss (Solaris) and strace (Linux); system activity monitors such as FileMon, RegMon, Process Monitor, and other Sysinternals utilities (Windows); and sniffers and protocol analyzers that monitor network traffic.

Attach the monitor to the process and look for library functions that indicate when randomness is being used. Run the process multiple times to see if the seed changes. Look for accesses of devices or equivalent resources that are commonly used for strong (or weak) randomness, such as /dev/urandom on Linux. Look for library or system calls that access predictable information such as process IDs and system time.

Background Details

Computers are deterministic machines, and as such are unable to produce true randomness. Pseudo-Random Number Generators (PRNGs) approximate randomness algorithmically, starting with a seed from which subsequent values are calculated. There are two types of PRNGs: statistical and cryptographic. Statistical PRNGs provide useful statistical properties, but their output is highly predictable and forms an easy to reproduce numeric stream that is unsuitable for use in cases where security depends on generated values being unpredictable. Cryptographic PRNGs address this problem by generating output that is more difficult to predict. For a value to be cryptographically secure, it must be impossible or highly improbable for an attacker to distinguish between it and a truly random value.

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	254	Security Features	**Development Concepts (primary)699** **Seven Pernicious Kingdoms (primary)700**
ChildOf	Category	723	OWASP Top Ten 2004 Category A2 - Broken Access Control	**Weaknesses in OWASP Top Ten (2004) (primary)711**
ChildOf	Category	747	CERT C Secure Coding Section 49 - Miscellaneous (MSC)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
ChildOf	Category	753	2009 Top 25 - Porous Defenses	**Weaknesses in the 2009 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)750**
ChildOf	Category	808	2010 Top 25 - Weaknesses On the Cusp	**Weaknesses in the 2010 CWE/SANS Top 25 Most Dangerous Programming Errors (primary)800**
ParentOf	Weakness Variant	329	Not Using a Random IV with CBC Mode	**Research Concepts (primary)1000**
ParentOf	Weakness Base	331	Insufficient Entropy	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	334	Small Space of Random Values	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Class	335	PRNG Seed Error	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	338	Use of Cryptographically Weak PRNG	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Class	340	Predictability Problems	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	341	Predictable from Observable State	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	342	Predictable Exact Value from Previous Values	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	343	Predictable Value Range from Previous Values	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	344	Use of Invariant Value in Dynamically Changing Context	**Development Concepts (primary)699** **Research Concepts (primary)1000**
ParentOf	Weakness Base	804	Guessable CAPTCHA	Development Concepts699 Research Concepts1000
MemberOf	View	1000	Research Concepts	**Research Concepts (primary)1000**

Relationship Notes

This can be primary to many other weaknesses such as cryptographic errors, authentication errors, symlink following, information leaks, and others.

Functional Areas

- Non-specific
- Cryptography
- Authentication

- Session management

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
PLOVER			Randomness and Predictability
7 Pernicious Kingdoms			Insecure Randomness
OWASP Top Ten 2004	A2	CWE More Specific	Broken Access Control
CERT C Secure Coding	MSC30-C		Do not use the rand() function for generating pseudorandom numbers
WASC	11		Brute Force
WASC	18		Credential/Session Prediction

Related Attack Patterns

CAPEC-ID	Attack Pattern Name	*(CAPEC Version: 1.5)*
59	Session Credential Falsification through Prediction	
112	Brute Force	
281	Analytic Attacks	

References

J. Viega and G. McGraw. "Building Secure Software: How to Avoid Security Problems the Right Way". 2002.

[REF-11] M. Howard and D. LeBlanc. "Writing Secure Code". Chapter 8, "Using Poor Random Numbers" Page 259. 2nd Edition. Microsoft. 2002.

Content History

Submissions

Submission Date	Submitter	Organization	Source
	PLOVER		Externally Mined

Modifications

Modification Date	Modifier	Organization	Source
2008-07-01	Eric Dalci updated Time of Introduction	Cigital	External
2008-09-08	CWE Content Team updated Background Details, Relationships, Other Notes, Relationship Notes, Taxonomy Mappings, Weakness Ordinalities	MITRE	Internal
2008-11-24	CWE Content Team updated Relationships, Taxonomy Mappings	MITRE	Internal
2009-01-12	CWE Content Team updated Description, Likelihood of Exploit, Other Notes, Potential Mitigations, Relationships	MITRE	Internal
2009-03-10	CWE Content Team updated Potential Mitigations	MITRE	Internal
2009-05-27	CWE Content Team updated Demonstrative Examples, Related Attack Patterns	MITRE	Internal
2009-12-28	CWE Content Team updated Applicable Platforms, Common Consequences, Description, Observed Examples, Potential Mitigations, Time of Introduction	MITRE	Internal
2010-02-16	CWE Content Team updated References, Relationships, Taxonomy Mappings	MITRE	Internal
2010-04-05	CWE Content Team updated Related Attack Patterns	MITRE	Internal

Previous Entry Names

Change Date	Previous Entry Name
2008-04-11	Randomness and Predictability

BACK TO TOP

Jailbrake File Referenced By Name

Risk

What might happen

In a jail-broken device, an attacker could manipulate the contents of a file written by the application. A buffer overflow or other unintended behavior may happen when the modified file is read back by the application. This may allow an attacker to seize control of the system.

Cause

How does it happen

The application creates a temporary file. An attacker detects a file creation event, deletes the created file, and creates a new file with the same name but different permissions. The application writes to a file referencing it by name. Data is written to the file owned by the attacker without the application noticing it. The attacker steals sensitive information from the temporary file or modifies its content. The application reads back the tampered file, and a buffer overflow or other unintended behavior happens. The attacker exploits the buffer overflow to gain control of the application.

For more information see section “Files in Publicly Writable Directories Are Dangerous” in “Apple Secure Coding Guide” document: https://developer.apple.com/library/ios/documentation/Security/Conceptual/SecureCodingGuide/

General Recommendations

How to avoid it

Reference all files by descriptor rather than by name.

Source Code Examples

Objective-C

Example of insecure way to reference files by name.

```
NSString* fileName = [NSTemporaryDirectory() stringByAppendingString:@"/f1.txt"];


NSString* stringToWrite = @"Hello world";
[stringToWrite writeToFile:fileName atomically:YES];
```

Example of secure way to reference files by descriptor.

```
NSString* fileNameTemplate = [NSTemporaryDirectory()
stringByAppendingString:@"/myTmpFile-XXXXXX.txt"];

const size_t bufLength = 2048;

char* fileNameBuf[bufLength];
strncpy(fileNameBuf, [fileNameTemplate cStringUsingEncoding:NSUTF8StringEncoding],
bufLength);

int fileDescriptor = mkstemp(fileNameBuf);

NSFileHandle* fileHandle = [[NSFileHandle alloc]
initWithFileDescriptor:fileDescriptor];


NSString* stringToWrite = @"Hello world";
[fileHandle writeData: [stringToWrite dataUsingEncoding:NSUTF8StringEncoding]];
```

Jailbreak Unchecked File Operation Result Code

Risk

What might happen

Data written to temporary file may become corrupted. A buffer overflow or other unintended behavior may happen when the corrupted file is read back by the application. This may allow an attacker to seize control of the system.

Cause

How does it happen

The application writes data to a temporary file. The write operation fails in the middle. Part of the data is written to the file, and another part is lost. The application does not check the result code of the file operation, and takes no corrective actions. The application reads back the corrupted file, and a buffer overflow or other unintended behavior happens. An attacker exploits the buffer overflow to gain control of the application.

For more information see section "Check Result Codes" in "Apple Secure Coding Guide" document: https://developer.apple.com/library/ios/documentation/Security/Conceptual/SecureCodingGuide/

General Recommendations

How to avoid it

Check the result code of every file operation and address failures appropriately.

Source Code Examples

Objective-C
Example of ignoring file operation result code (insecure).

```
NSString* stringToWrite = @"Hello world";

NSString* filePath = [NSTemporaryDirectory() stringByAppendingString:@"/f1.txt"];

[stringToWrite writeToFile:filePath atomically:NO];
```

Example of checking file operation result code (secure).

```
NSString* stringToWrite = @"Hello world";

NSString* filePath = [NSTemporaryDirectory() stringByAppendingString:@"/f1.txt"];

if (![stringToWrite writeToFile:filePath atomically:NO])
{
    [[NSException exceptionWithName:@"FileWriteError" reason:@"" userInfo:nil]
raise];
};
```

Unchecked CString Convertion

Risk

What might happen

Converted C-String length may be less than the CFString length. If the program then makes decisions based on that erroneously converted string, any number of erroneous behaviors can result.

Cause

How does it happen

A CFString have an explicit length and can contain null bytes at arbitrary locations in the data. A CFString with a null character in the middle is converted into C-String. The resulting C-String is evaluated. The application behaves incorrectly behavior because the resulting C string effectively ends at the first null byte.

Example of a real life attack:

This vulnerability occurred in many SSL stacks a few years ago. By applying for an SSL cert for a carefully crafted subdomain of a domain that you own, you could effectively create a certificate that was valid for arbitrary domains.

Consider a subdomain in the form targetdomain.tld[null_byte].yourdomain.tld.

Because the certificate signing request contains a Pascal string (which like CFString can contain null characters), assuming that the certificate authority interprets it correctly, the certificate authority would contact the owner of yourdomain.tld and would ask for permission to deliver the certificate. Because you own the domain, you would agree to it. You would then have a certificate that is valid for the rather odd-looking subdomain in question.

When checking the certificate for validity, however, many SSL stacks incorrectly converted that Pascal string into a C string without any validity checks. When this happened, the resulting C string contained only the targetdomain.tld portion. The SSL stack then compared that truncated version with the domain the user requested, and interpreted the certificate as being valid for the targeted domain.

In some cases, it was even possible to construct wildcard certificates that were valid for every possible domain in such browsers (*.com[null] .yourdomain.tld would match every .com address, for example).

For more information see section “Avoiding Buffer Underflows” in “Apple Secure Coding Guide” document:
https://developer.apple.com/library/ios/documentation/Security/Conceptual/SecureCodingGuide/

General Recommendations

How to avoid it

1. Avoid converting non-C strings (CFStringRef objects, NSString objects, CFDataRef objects) into C strings if possible. Instead, work with the strings in their original format.
2. If this is not possible, always perform length checks on the resulting C string or check for null bytes in the source data.

Source Code Examples

Objective-C
Example of converting CFString to C-String without length check (insecure).

```
CFStringRef s1 = CFSTR("Hello");

size_t buf_len = 10;

char buffer[buf_len];
CFStringGetCString(s1,buffer,buf_len,kCFStringEncodingASCII);
```

Failure to Clear Heap Memory Before Release ('Heap Inspection')

Weakness ID: 244 *(Weakness Variant)* **Status:** Draft

Description

Description Summary

Using realloc() to resize buffers that store sensitive information can leave the sensitive information exposed to attack, because it is not removed from memory.

Extended Description

When sensitive data such as a password or an encryption key is not removed from memory, it could be exposed to an attacker using a "heap inspection" attack that reads the sensitive data using memory dumps or other methods. The realloc() function is commonly used to increase the size of a block of allocated memory. This operation often requires copying the contents of the old memory block into a new and larger block. This operation leaves the contents of the original block intact but inaccessible to the program, preventing the program from being able to scrub sensitive data from memory. If an attacker can later examine the contents of a memory dump, the sensitive data could be exposed.

Time of Introduction

- Implementation

Applicable Platforms

Languages

C

C++

Common Consequences

Scope	Effect
Confidentiality	Be careful using vfork() and fork() in security sensitive code. The process state will not be cleaned up and will contain traces of data from past use.

Demonstrative Examples

Example 1

The following code calls realloc() on a buffer containing sensitive data:

(Bad Code)

Example Language: ***C***

```
cleartext_buffer = get_secret();...
cleartext_buffer = realloc(cleartext_buffer, 1024);
...
scrub_memory(cleartext_buffer, 1024);
```

There is an attempt to scrub the sensitive data from memory, but realloc() is used, so a copy of the data can still be exposed in the memory originally allocated for cleartext_buffer.

Relationships

Nature	Type	ID	Name	View(s) this relationship pertains to
ChildOf	Weakness Base	226	Sensitive Information Uncleared Before Release	**Research Concepts (primary)1000**
ChildOf	Weakness Class	227	Failure to Fulfill API Contract ('API Abuse')	**Development Concepts (primary)699** **Seven Pernicious Kingdoms (primary)700**
ChildOf	Category	633	Weaknesses that Affect Memory	**Resource-specific Weaknesses (primary)631**
ChildOf	Category	742	CERT C Secure Coding Section 08 - Memory Management (MEM)	**Weaknesses Addressed by the CERT C Secure Coding Standard (primary)734**
CanPrecede	Weakness Class	669	Incorrect Resource Transfer Between Spheres	Research Concepts1000
MemberOf	View	630	Weaknesses Examined by SAMATE	**Weaknesses Examined by SAMATE (primary)630**

Affected Resources

- Memory

Taxonomy Mappings

Mapped Taxonomy Name	Node ID	Fit	Mapped Node Name
7 Pernicious Kingdoms			Heap Inspection
CERT C Secure Coding	MEM03-C		Clear sensitive information stored in reusable resources returned for reuse

White Box Definitions

A weakness where code path has:

1. start statement that stores information in a buffer
2. end statement that resize the buffer and
3. path does not contain statement that performs cleaning of the buffer

Content History

Submissions			
Submission Date	**Submitter**	**Organization**	**Source**
	7 Pernicious Kingdoms		Externally Mined

Modifications			
Modification Date	**Modifier**	**Organization**	**Source**
2008-08-01		KDM Analytics	External
	added/updated white box definitions		
2008-09-08	CWE Content Team	MITRE	Internal
	updated Applicable Platforms, Name, Relationships, Other Notes, Taxonomy Mappings		
2008-10-14	CWE Content Team	MITRE	Internal
	updated Relationships		
2008-11-24	CWE Content Team	MITRE	Internal
	updated Relationships, Taxonomy Mappings		
2009-05-27	CWE Content Team	MITRE	Internal
	updated Demonstrative Examples, Name		
2009-10-29	CWE Content Team	MITRE	Internal
	updated Common Consequences, Description, Other Notes		

Previous Entry Names	
Change Date	**Previous Entry Name**
2008-04-11	Heap Inspection
2008-09-09	Failure to Clear Heap Memory Before Release
2009-05-27	Failure to Clear Heap Memory Before Release (aka 'Heap Inspection')

BACK TO TOP

Buffer Size Literal Condition

Risk

What might happen

An attacker can exploit the buffer overflow to execute an arbitrary code with the privileges of the vulnerable application.

Cause

How does it happen

Buffer size is specified by a number literal, and access to the buffer is checked using number literal. A developer make changes to the core, and reduces the allocated buffer size, but forgets to reduce the literal in conditional statement. A buffer overflow is created.

At run time, the following scenario happens: The overflown buffer is allocated in stack. Right after the end of the buffer, the function return address is located. An attacker manipulates the input in such a way that when data is written into the buffer the return address is overwritten. The new return address points to a memory segment under the attacker control. When the function returns, the attacker code is executed with the application privileges.

General Recommendations

How to avoid it

Define a constant that holds the buffer size, then use this constant throughout the code.

Source Code Examples

Objective-C
Code example that is vulnerable to buffer overflow after code changes.

```
void f() {
     char buf[10];
     char* sourceString = istream.read("input");
     if (strlen(sourceString) < 10)
     {
         strcpy(buf,sourceString);
     }
}
```

Code that is resilient to buffer overflow.

```
void f()
{
    const int MAX_INPUT_SIZE = 256;
    const int BUFFER_SIZE = 10;
    char buf[BUFFER_SIZE];
    char* sourceString = istream.read("input");
    if (strnlen(sourceString,MAX_INPUT_SIZE) < BUFFER_SIZE)
    {
        strcpy(buf,sourceString);
    }
}
```