

Switch from Unsafe Serialization API to NSCoder

🍏 CultureNEXT - PreProd 1.0.0 ID \$100814 Severity: Medium No Priority

The following binaries within the App contain code which leverages `NSCoding`, which is vulnerable to object substitution attacks:

OPEN	NSCoding protocol used in binary <i>cultureNEXT.app/cultureNEXT</i> since 4/13/2021	Stack Trace ▾
NOT FOUND	NSCoding protocol used in binary <i>CultureNEXT.app/CultureNEXT</i> since 4/12/2021	Stack Trace ▾

DESCRIPTION

NSCoding is an Objective-C protocol designed to allow serialization and deserialization of code objects. However, this protocol does not verify the type of object upon deserialization. Thus, it is vulnerable to object substitution attacks. A maliciously crafted payload which is deserialized via the NSCoder protocol can result in attacker-controlled code being executed.

Apple provides the [NSSecureCoding protocol](#) which is robust against this type of attack. NSSecureCoding protects against object substitution attacks by requiring the programmer to declare the expected type of object before deserialization completes. Thus, if an invalid object is deserialized, the error can be handled safely.

SCREENSHOT

The screenshot shows the Xcode interface with a search for "NSCoding" in the RoboRogue project. The search results list 100 results in 45 files. The selected file is RRGRoom.h, which contains the following code:

```

1 //
2 // RRGRoom.h
3 // RoboRogue
4 //
5 // Created by 山本政徳 on 2014/03/01.
6 // Copyright (c) 2014年 山本政徳. All rights reserved.
7 //
8
9 #import <Foundation/Foundation.h>
10
11 typedef NS_ENUM(NSUInteger, RRGRoomType)
12 {
13     RRGRoomTypeNormal,
14     RRGRoomTypeUnused,
15     RRGRoomTypeMonstersNest,
16     RRGRoomTypeShop,
17 };
18
19 @interface RRGRoom : NSObject <NSCoding>
20 @property (nonatomic) RRGRoomType roomType;
21 @property (nonatomic) CGRect roomRect;
22 @property (nonatomic) NSUInteger roomNum;
23
24 -(NSInteger)roomLX;
25 -(NSInteger)roomLY;
26 -(NSInteger)roomHX;
27 -(NSInteger)roomHY;
28 -(NSInteger)roomWidth;
29 -(NSInteger)roomHeight;
30
31 +(instancetype)roomWithRect:(CGRect)rect
32     roomType:(RRGRoomType)roomType;
33
34 -(void)addGateOut:(CGPoint)newGateOut
35     gateIn:(CGPoint)newGateIn;
36 -(CGPoint)exitGateOutAtRandom:(CGPoint)entranceGateOut;
37 @end
38

```

RECOMMENDATION

Locate all the classes in the App that conform to NSCoder and migrate them to NSSecureCoding. You can utilize Xcode's built-in search function to locate these classes in the App's project. Searching for "NSCoding" will reveal everything that conforms to the vulnerable protocol.

Additionally, ensure all input data is validated before it is used, especially when dealing with data that becomes executable.

You can read more about NSSecureCoding on [NSHipster](#).

SECURE CODE

```

// Declare that your class conforms to NSSecureCoding
@interface MySecureObject : NSObject <NSSecureCoding>
@property (nonatomic, retain) NSDictionary *myData;
@end

@implementation MySecureObject
+ (BOOL)supportsSecureCoding {
    // Must override this class delegate method to return YES
    return YES;
}
- (id)initWithCoder:(NSCoder *)decoder {
    if ((self = [super initWithCoder:decoder])) {
        // When decoding sub-objects, use @selector(decodedObjectOfClass:forKey:)
        // This method will throw an exception if the deserialized object's class doesn't match the expected class
        self.myData = [decoder decodedObjectOfClass:[NSDictionary class] forKey:@"myData"];
    }
    return self;
}
- (void)encodeWithCoder:(NSCoder *)encoder {
    [encoder encodeObject:self.myData forKey:@"myData"];
}
@end

```