datatheorem VIEW IN PORTAL

Switch from Unsafe Serialization API to NSSecureCoding

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 since 4/13/2021 `NSCoding` protocol used in binary *cultureNEXT.app/cultureNEXT*
Stack Trace

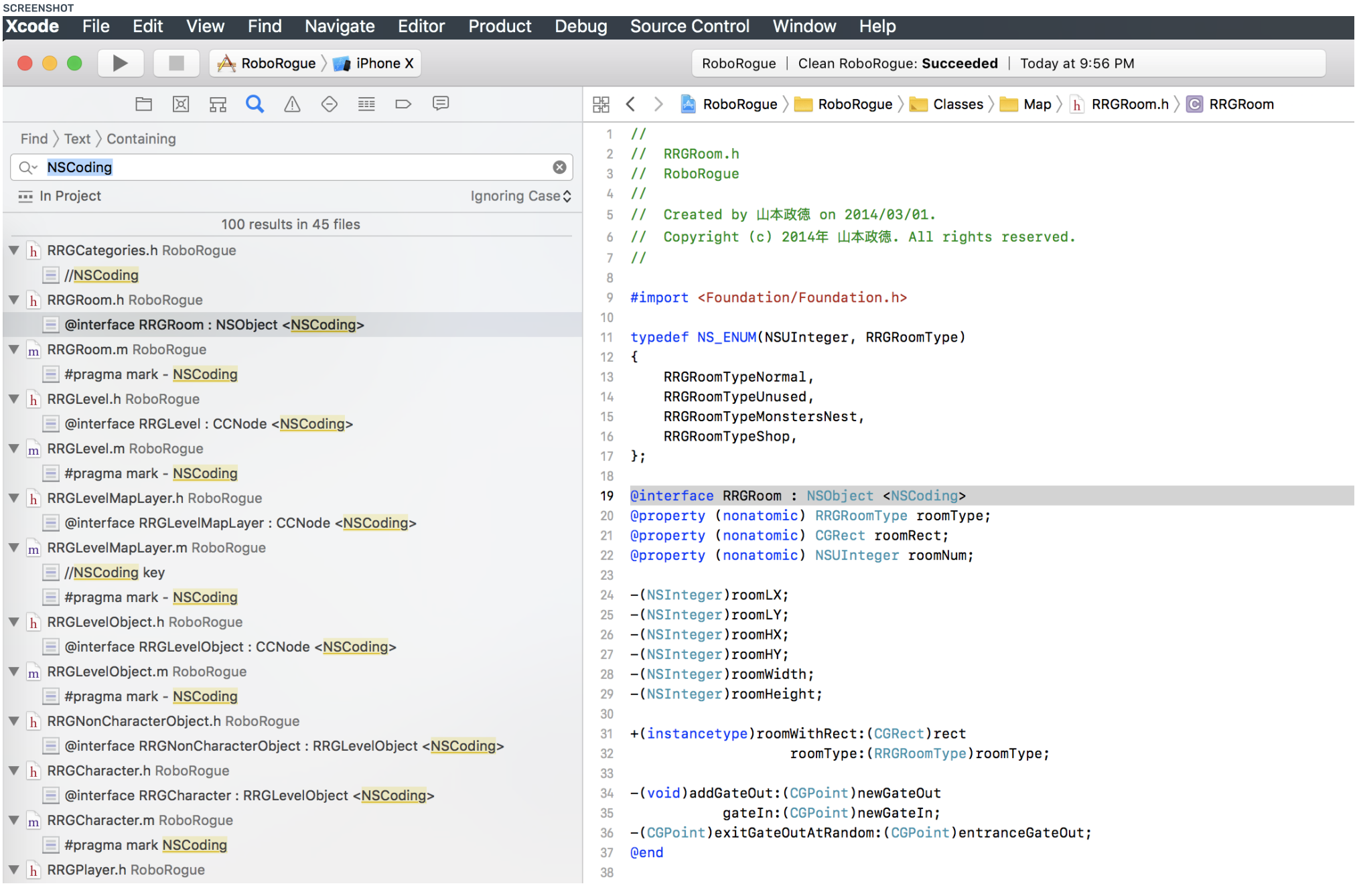
NOT FOUND since 4/12/2021 `NSCoding` protocol used in binary *CultureNEXT.app/CultureNEXT*
Stack Trace

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 `NSCoding` 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



RECOMMENDATION

Locate all the classes in the App that conform to `NSCoding` 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 reture YES
    return YES;
}
- (id)initWithCoder:(NSCoder *)decoder {
    if ((self = [super init])) {
        // When decoding sub-objects, use @selector(decodeObjectOfClass:forKey:)
        // This method will throw an exception if the deserialized object's class doesn't match the expected class
        self.myData = [decoder decodeObjectOfClass:[NSDictionary class] forKey:@"myData"];
    }
    return self;
}
- (void)encodeWithCoder:(NSCoder *)encoder {
    [encoder encodeObject:self.myData forKey:@"myData"];
}
@end
```