CodeCluedo: The findings

1. Line 01: Flask has a default XSS vulnerability because it returns input as a HTML string;
2. Line 04: MD5 is insecure hashing algorithm;
3. Line 08: Key is not random, and hardcoded;
4. Line 08: (Random) key is known by untrusted third party;
5. Line 09: /tmp is not secure (public, no POSIX authorizations can be set), but might also be cleared on reboot;
6. Line 12: ECB(DES) is an insecure algorithm and mode;
7. Line 12: Initiation Vector is static and therefore not random (0000000);
8. Line 12: No padding, which could have increased randomness;
9. Line 12: Only first 8 characters of random key are used, strongly reducing strength of encryption;
10. Line 13: No input check which could lead to path traversal and disk/performance issues;
11. Line 14: Limited path traversal, i.e. attacker can write files with "../" in filename in arbitrary directory;
12. Line 18: List_secure_data does not check input which could lead to:
    - Path/directory traversal
    - Command execution
13. Line 18: Remote Code Execution possible via "Path" parameter;
14. Line 18: Path traversal via path parameter;
15. Line 26: The Social Security Number(BSN) could contain HTML;
16. Line 27: BSN is stored plain tekst;
17. Line 27: BSN is stored, no consent is being asked to the user (legal/privacy issue);
18. Line 27: Modification possible with GET Parameter;
19. General: No explicit Cross Site Request Forgery protection;
20. General: No authentication/authorization of user;
21. General: No error/exception handling.

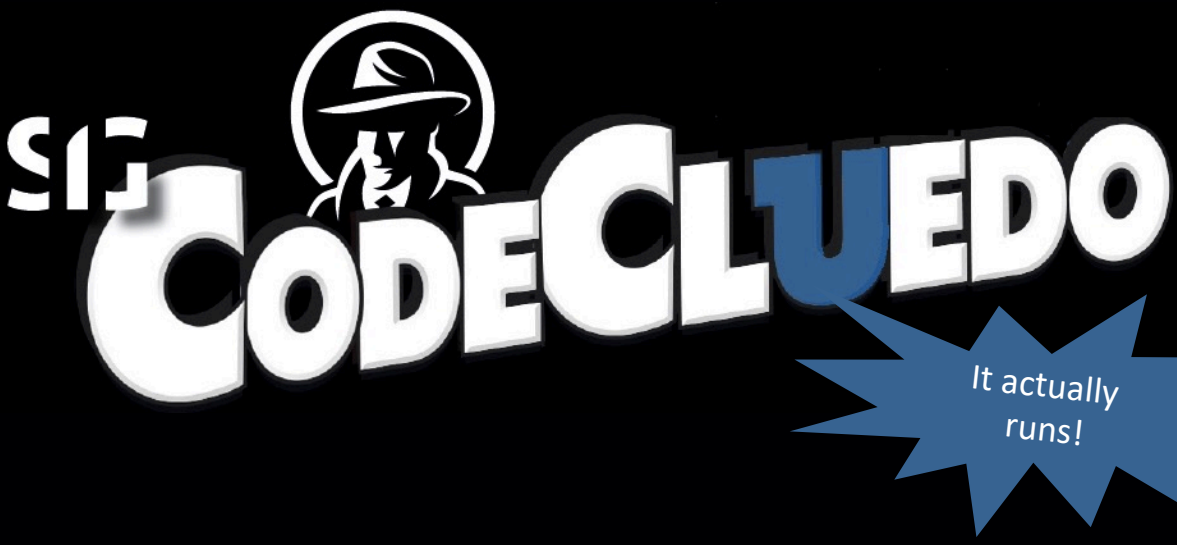
You want to know whodunnit? Who is guilty?
Well, if you look here, you see in the comments that Jo from marketing has requested socials security number to be stored, using a clever trick, by claiming that it is just needed for the demo and then we will take it out. Yeah right. Jo here is manipulating developers, because of course the business will like the demo. But there's more.
You can also see that John, the product owner has called late at night that marketing wants the team to rush in an insecure feature because we can't wait for a secure solution to be provided by IT. The third clue is that Tim from IT has built in a backdoor, by making sure that they created an encryption key to be hardcoded. And Tim is also Jo's husband. Why is Tim double-crossing the success of product owner John and marketeer Jo, and how does John learn late at night that marketing cannot wait? You get the picture?

This can only mean that John is sleeping with Jo, and that Jo is using John to get her way, and that Tim found out and is planning revenge on them both by sabotaging their product. You could say that Jo is guilty to manipulate John. You could say Tim is guilty for wanting revenge, you could say John is guilt for being manipulated, you could say the developers are guilty because they implemented this mess.
It comes down to this, the correct answer to who done it, is: everybody, everywhere with everything because they could get away with it. And because security is everybody's problem!

If you want to know more about the Software Improvement Group follow this link: www.sig.eu/security.

Please follow us if you want to keep up to date about Getting Software RIGHT:

Sharing of this document is allowed, but not partially.




```python
from flask import Flask, url_for, json, request
from pyDes import des
import commands
import md5
import pyDes

app = Flask(__name__)
RANDOM_KEY = md5.new("085ZMVsBnTYuO6OK7gfJpGXeik5VZamC").digest(); # Tim (Jo's husband,from IT) created random key
SECURE_DIRECTORY = '/tmp/' # John from Products called last night: marketing can't wait for IT's new directory

def secure_store(filename, suffix, data):
    IV = b"\0\0\0\0\0\0\0\0";
    d = des(RANDOM_KEY[0:8],pyDes.ECB, IV, pad=None, padmode=pyDes.PAD_PKCS5)
    f = open(SECURE_DIRECTORY + '/' + filename + '-' + suffix, 'w')
    f.write(d.encrypt(bytes(data)))
    f.close()
    return 'data stored'

def list_secure_data(path): return commands.getstatusoutput('ls ' + SECURE_DIRECTORY + '/' + path)[1]

@app.route('/')
def api_root(): return 'Welcome to employee data storage api'

@app.route('/employee')
def api_employee():
    # Jo(Marketing) needs social security nr temporarily in next demo
    s = {"list": lambda: list_secure_data(request.args['ssn']),
         "add": lambda: secure_store(request.args['ssn'], request.args['email'],
             request.args['data'])}
    return s.get(request.args['cmd'], lambda: "no such command")()

if __name__ == '__main__': app.run()
```

Software Improvement Group (SIG) analyses code for weaknesses and then diagnoses these to find root causes: much like a game of Cluedo. Whodunnit? Was it the CEO with the tight deadline in the board room, or was it the lead developer with the selfmade crypto in the home office?

Sharing of this document is allowed, but not partially.