TDT4237 2024 Sensor guide

The letter grade follows the NTNU grading scale using percentage points (Grading scale using percentage points - Knowledge base - NTNU**)**

A: 89–100 points

B: 77–88 points

C: 65–76 points

D: 53–64 points

E: 41–52 points

F: 0–40 points

Answers to questions and corresponding grading guidelines

Q2: Strategy to mitigate the security compromise impact (4 points)

Suppose your system takes users' input and can be exposed to injection attacks. List and explain at least four strategies to mitigate the impact of injection attack compromises. (4 points)

Any four of the following answers shall be ok to get the four points.

Avoid information leakage
– Don’t display a detailed error message to external users
– Don’t display stack traces to external users

Limiting privileges
– No more privileges than users need – E.g., Read access on tables/views the user can query – E.g., No drop table privilege for a typical user

Encrypt sensitive data, e.g.,
– Username, credit card number, magical powers

Key management precautions – Do not store the encryption key in DB

Hash password

The other valid answers are:

Blacklisting, Whitelisting, Escaping, Prepared statements, and meaningful countermeasures.

Q3: Question undefinedSecurity and Integrity of OTP (4 points)
1. Explain encryption and decryption algorithm of One Time Pad (OTP). (1 point)

2. Explain why it is insecure to use the same key to encrypt two or several messages using OTP. (2 points)

3. Explain why OTP cannot guarantee integrity (1 point)

1. The One Time Pad (OTP) is a symmetric encryption scheme that uses a random key of the same length as the plaintext to encrypt and decrypt messages:

- Generate a random key of the same length as the plaintext.
- Convert the plaintext and the key to binary form.
- Perform a bitwise XOR operation between the plaintext and the key to obtain the ciphertext.
- Convert the ciphertext to a human-readable format.

The decryption algorithm is the same as the encryption algorithm, except that it uses the same key to perform the XOR operation between the ciphertext and the key to obtain the plaintext.

2. It is insecure to use the same key to encrypt two or several messages using OTP because it violates the "one-time" property of the OTP. If the same key is used twice, an attacker can use the XOR operation between two ciphertexts to obtain the XOR operation between the two plaintexts, which may reveal some information about the plaintexts. If the attacker knows or can guess part of one of the plaintexts, she can use that information to obtain part of the other plaintext.
3. If the attack modifies the cyphertext, it is impossible for the message receiver to notice that if the information after decryption still seems to make sense. Thus, OTP cannot guarantee integrity.

For Q3.1 and Q3.2, if the XOR is not mentioned, no point shall be given.

Q4: AI and security (4 points)

What is the difference between malicious abuse of AI and malicious use of AI? Give examples of both.

Malicious abuse of AI: when AI is the attack target, exploiting its vulnerabilities. Manipulation of AI algorithms and training data to cause the AI system behave incorrectly.

Example: A common example is where you fool a ML classifier by adding some kind of pertubation into an image, leading to misclassification of the image.

Malicious use of AI: when AI is used to enhance attacks against any type of system. Using AI to improve the speed, scale, and sophistication of attacks.

Example: advanced social engineering and targeted phishing attacks, use of AI for evading detection and authentication controls, voice imitation to gather sensitive data for banking transactions using AI-supported voice synthesis.

Q5: Privacy-related question (4 points)

How can a controller demonstrate data protection? Give at least 5 examples.

Any four of the following will get four points.

logs of static code analysis
threat models
trainings to developer
code review reports
security checklists
test logs
bug bounties
incident management plan

The other valid answers are GDPR principles, such as:

Lawfulness, fairness, and transparency; Purpose limitation; Data minimization, Accuracy; Storage limitation; Integrity and confidentiality; and Accountability.

Q6: Code to setup password policy (4 points)

Suppose the password policy of a system is as follows.

- The password is between 8-10 characters long
- The password contains characters from 3 of the following 4 categories:
    - standard uppercase characters (A - Z)
    - standard lowercase characters (a - z)
    - numbers (0 - 9)
    - symbols: only from among ! % - _ + = [ ] { } : , . ? < > ( ) ;
- The password **does not** contain information identical to user's first and last name
- The password **does not** contain common passwords
- **Spaces and the letters "æ", "ø" and "å" are not accepted**

The 8-10 characters long is not properly checked (- 1 point)

The common password or user name is not properly checked (-1 point)

Not properly implemented or mentioned the following policy (-1 point) or the validators are significantly wrong (-1 point)

The password contains characters from 3 of the following 4 categories:

- standard uppercase characters (A - Z)
- standard lowercase characters (a - z)
- numbers (0 - 9)
- symbols: only from among ! % - _ + = [ ] { } : , . ? < > ( ) ;

Q7: Circuit breaker pattern (4 points)

What is the purpose of the circuit breaker pattern in the context of microservice architecture security?

Services are monitored, if a service is not working, the number of failures will be counted. If the number of failures passes some threshold, a fallback method will be invoked.

If answers show that students know the purpose is to prevent failure/attack propagation or fail-safe, they can get full points.

Q8: Question undefined Authorization (4 points)

1) In the Bell-LaPadula model, what does the * property mean?

2) What about STRONG * ?

1) A subject cannot write to a lower classification
2) Subjects may only write to objects with the matching security level

We deduct two points if any of the two parts is wrong. For the second one, if the student mentions that it means both Biba and Bell-LaPadula shall be satisfied, it is also a valid answer.

Q9: Concept drift (4 points)

What is the concept drift challenge within AI? Give an example.

Concept drift refers to the phenomenon where the statistical properties of the data change over time, leading to potential degradation in the model's performance as it continues to make predictions based on outdated patterns.

This is especially relevant in dynamic environments, where underlying data distribution can evolve over time, and thus security properties may degrade.

The environment constantly changes, preventing the reliable application of ML in the long term because the training data can quickly become obsolete.

Example: an AI model used for predicting consumer behavior might become less effective as consumers may change their behaviors over time, leading to the need to retrain the model to adapt to new patterns.

Q10: Penetration Testing (4 points)

What are the pros and cons of penetration testing tools?

Pro:

- Helpful – but very important to know what they do and don't do
- Frameworks are getting better, automated scanners are becoming less effective
- Saves time

Cons:

- Have limited coverage
- Can provide a false sense of security if not fully understood
- Expensive

Other good suggestions are accepted as well.

Q11: Social Engineering (4 points)

Mention principles of persuasion that can be used for social engineering attacks.

PEOPLE GIVE when they get
PEOPLE LISTEN to authority
PEOPLE DO as similar people do
PEOPLE LIKE those who like them
PEOPLE COMMIT to their statements
PEOPLE AVOID loss of advantage

Students do not need to list all of them. Each of the above can get one point. Any four of the above can get all points.

Security logging vulnerability fixing (4 points)

production.py

```
1 from app.settings.common import *
2 from decouple import config, Csv
3
4
5 # Email admins and managers
6 # https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/#admins-and-
managers
7 ADMINS = config('ADMINS', default=[('Admin',
'admin@forum.securecodewarrior.com')], cast=Csv())
8 MANAGERS = config('MANAGERS', default=[('moderator',
'moderator@forum.securecodewarrior.com')], cast=Csv())
9
10
11 # SECURITY WARNING: don't run with debug turned on in production!
12 DEBUG = False
13
14 # https://docs.djangoproject.com/en/2.1/ref/settings/#allowed-hosts
15 ALLOWED_HOSTS = ['forum.securecodewarrior.com', ]
16
17
18 # SSL/HTTPS
19 # https://docs.djangoproject.com/en/2.1/topics/security/#ssl-https
20
21 SECURE_SSL_REDIRECT = True
22
23 SESSION_COOKIE_SECURE = True
24
25 # Cross Site Request Forgery
26 CSRF_COOKIE_SECURE = True
27 CSRF_TRUSTED_ORIGINS = config('CSRF_TRUSTED_ORIGINS', default=[],
cast=Csv())
28
29 SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
```

```
30
31
32 # https://docs.djangoproject.com/en/2.1/topics/security/
33 # HSTS
34 SECURE_HSTS_SECONDS = 5 * 60
35 SECURE_HSTS_PRELOAD = True
36 SECURE_HSTS_INCLUDE_SUBDOMAINS = True
37
38 # https://docs.djangoproject.com/en/2.1/ref/clickjacking/
39 X_FRAME_OPTIONS = 'DENY'
40
41 SECURE_CONTENT_TYPE_NOSNIFF = True
42
43 SECURE_BROWSER_XSS_FILTER = True
44
45 # Sessions
46 # https://docs.djangoproject.com/en/2.1/ref/settings/#sessions
47 SESSION_COOKIE_AGE = 24 * 60 * 60  # 24 hours, in seconds
48 SESSION_EXPIRE_AT_BROWSER_CLOSE = True
49
50
51 # Database
52 # https://docs.djangoproject.com/en/2.1/ref/settings/#databases
53
54 DATABASES = {
55     'default': {
56         'ENGINE': 'django.db.backends.postgresql',
57         'NAME': config('DATABASE_NAME'),
58         'USER': config('DATABASE_USER'),
59         'PASSWORD': config('DATABASE_PASSWORD'),
60         'HOST': config('DATABASE_HOST'),
61         'PORT': config('DATABASE_PORT'),
62     }
63 }
64
65
66 # Email configuration
67 # https://docs.djangoproject.com/en/2.1/ref/settings/#default-from-email
68
69 DEFAULT_FROM_EMAIL = 'support@forum.securecodewarrior.com'
70 SERVER_EMAIL = 'admin@forum.securecodewarrior.com'
71
72
73 # https://docs.djangoproject.com/en/2.1/ref/settings/#email-backend
74 EMAIL_BACKEND = config('EMAIL_BACKEND')
75 EMAIL_HOST = config('EMAIL_HOST')
76 EMAIL_PORT = config('EMAIL_PORT', cast=int)
77 EMAIL_USE_TLS = config('EMAIL_USE_TLS', cast=bool)
78 EMAIL_HOST_USER = config('EMAIL_HOST_USER')
79 EMAIL_HOST_PASSWORD = config('EMAIL_HOST_PASSWORD')
```

```
80
81
82 # https://docs.djangoproject.com/en/2.1/ref/settings/#file-upload-permissions
83 FILE_UPLOAD_PERMISSIONS = config('FILE_UPLOAD_PERMISSIONS',
default=0o640, cast=oct)
```

Code snippet of common.py

```
1 # Logging
2 # https://docs.djangoproject.com/en/2.1/topics/logging/#configuring-logging
3
4 # Disable Django's logging setup
5 LOGGING_CONFIG = None
6
7 LOGLEVEL = config('LOGLEVEL', default='INFO')
8
9 # https://docs.djangoproject.com/en/2.1/topics/logging/#custom-logging-configuration
10 logging.config.dictConfig({
11     'version': 1,
12     'disable_existing_loggers': False,
13     'formatters': {
14         'default': {
15             # exact format is not important, this is the minimum information
16             'format': '%(asctime)s %(name)-12s %(levelname)-8s %(message)s',
17         },
18         'django.server': DEFAULT_LOGGING['formatters']['django.server'],
19     },
20     'handlers': {
21         # console logs to stderr
22         'console': {
23             'class': 'logging.StreamHandler',
24             'formatter': 'default',
25         },
26         'file': {
27             'level': 'WARNING',
28             'class': 'logging.FileHandler',
29             'filename': os.path.join(BASE_DIR, 'debug.log'),
30         },
31         'django.server': DEFAULT_LOGGING['handlers']['django.server'],
32     },
33     'loggers': {
34         # default for all undefined Python modules
35         '': {
36           'level': LOGLEVEL,
37             'handlers': ['console', 'file'],
38         },
39         # Prevent noisy modules from logging
40         'noisy_module': {
41             'level': 'ERROR',
42             'handlers': ['console'],
43             'propagate': False,
44         },
```

```
45         # Default runserver request logging
46         'django.server': DEFAULT_LOGGING['loggers']['django.server'],
47     },
48 })
```

The code snippet common.py has vulnerabilities related to security logging and monitoring. The code in production.py provides some background information.

- Explain which lines in common.py are vulnerable and why they are vulnerable. (2 points)

- Explain how to fix the vulnerabilities in the common.py. It is necessary to provide code to show how to fix it. (2 points) (Note: Syntax errors are allowed, especially if you explain the code.)

```
# https://docs.djangoproject.com/en/2.1/topics/logging/#custom-logging-configuration
logging.config.dictConfig({
    'version': 1,
    'disable_existing_loggers': False,
    'formatters': {
        'default': {
            # exact format is not important, this is the minimum information
            'format': '%(asctime)s %(name)-12s %(levelname)-8s %(message)s',
        },
        'django.server': DEFAULT_LOGGING['formatters']['django.server'],
    },
    'handlers': {
        # console logs to stderr
        'console': {
            'class': 'logging.StreamHandler',
            'formatter': 'default',
        },
        'file': {
            'level': 'WARNING',
            'class': 'logging.FileHandler',
            'filename': os.path.join(BASE_DIR, 'debug.log'),
        },
        'mail_admins': {
```

```
            'level': 'CRITICAL',
            'class': 'django.utils.log.AdminEmailHandler',
        },
        'django.server': DEFAULT_LOGGING['handlers']['django.server'],
    },
    'loggers': {
        # default for all undefined Python modules
        '': {
            'level': LOGLEVEL,
            'handlers': ['console', 'file', 'mail_admins'],
        },
        # Prevent noisy modules from logging
        'noisy_module': {
            'level': 'ERROR',
            'handlers': ['console'],
            'propagate': False,
        },
        # Default runserver request logging
        'django.server': DEFAULT_LOGGING['loggers']['django.server'],
    },
})
```

If students explain that the vulnerability is that there is no mechanism to inform the administrator/operator by email if any error/attack happens and explain briefly how to fix it, they will get full points.

Other issues are not relevant and do not get points.

Q13: Supply chain security (4 points)

What are the four steps of software supply chain attacks and what are the corresponding countermeasure strategies? (4 points)

Four steps:

- Compromise: First, an attacker finds and compromises an existing weakness within a supply chain.
- Alteration: Second, an attacker leverages the initial compromise to alter the software supply chain.

- Propagation: Third, the change introduced by the attacker propagates to downstream components and links.
- Exploitation: The attacker exploits the alterations in a downstream link.

Countermeasure strategies
- Transparency
- Validity
- Separation
- Recovery

We give each correct item 0.5 point.

Q14: Microservice architecture attack area (4 points)

What are potential attack areas of microservices deployed on a cloud? (4 points)

The attack areas include:
1) system entry point
2) load balancer
3) service entry point
4) communications between services
5) container or hardware

We give each correct item 1 point.

14 Microservice architecture attack area (4 points)

What are potential attack areas of microservices deployed on a cloud? (4 points)

Fill in your answer here

Format

Words: 0

15 Privacy motivation (1 Point)

What is the biggest motivation for software companies to work with privacy?

Select one alternative:

- Big legal fines ✔
- This is what management cares about
- This is what developers care about
- The respect of their customers
- Catching criminals

16 Cryptography keys (1 point)

Which of the following methods is NOT a recommended approach for generating cryptographic keys?

Select one alternative:

- Collecting entropy from user-generated input, such as mouse movements or keyboard strokes.
- Using a hardware random number generator
- Employing a software-based secure pseudo-random number generator with unique seeds
- Reusing a previously generated key for a new encryption task ✔
- Deriving keys from a passphrase using a key derivation function

17 Threat modeling (1 point)

What is the best way of performing threat modeling?

Select one alternative:

- You should create your own threat modeling technique that is tailored for the job.
- Attack trees were the first and is still the most recognized way of modeling threats.
- Misuse case diagrams were invented at NTNU and considered to be the most useful way.
- It is better to create multiple threat modeling representations because there is no single ideal view ✔
- DFD is the most widely used threat modeling technique and should therefore be used

18 Injection vulnerability in the code (1 point)

cart.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title> User Cart </title>
</head>
<body>
     {% for item in items %}

          Product Name: {{ item.product_name }}<br>
          Product Id : {{ item.product_id }}<br>
          Transaction Id : {{ item.transaction }}<br>
          Price : {{ item.price }}<br>

     {% endfor %}
<br><br><b><a href="{% url 'shops:logout' %}"> logout</a></b>
</body>
</html>
```

models.py

```
from __future__ import unicode_literals

from django.db import models
from django.contrib.auth.models import User


# Model for shopping order transactions.
class ShoppingTransaction(models.Model):
    order_id = models.CharField(max_length=20)
    date = models.DateTimeField()
    total_amount = models.DecimalField(max_digits=7, decimal_places=2)
    user = models.ForeignKey(User)

    def __str__(self):
        return self.order_id


# Model for products purchased by a customer.
class TransactionDetail(models.Model):
    product_id = models.CharField(max_length=20)
    product_name = models.CharField(max_length=40)
    transaction = models.ForeignKey(ShoppingTransaction)
    price = models.DecimalField(max_digits=7, decimal_places=2)
```

views.py

```
from django.shortcuts import render
from django.http import HttpResponseRedirect
from django.contrib.auth import authenticate, login, logout
from django.contrib.auth.decorators import login_required
from django.core.urlresolvers import reverse
from django.contrib import messages

from shops.models import TransactionDetail, ShoppingTransaction
from shops.forms import LoginForm


# User Login
def user_login(request):
    # Checking the request method
    if request.method == 'POST':
        # Create a form instance and populate it with data from the request:
        form = LoginForm(request.POST)
        # Checking if all the form fields value meet the set criteria
        if form.is_valid():
            # Fetching the username and passwords from POST methods
            user_name = form.cleaned_data['username']
            pass_word = form.cleaned_data['password']
            # Authenticating the user
            user = authenticate(username=user_name, password=pass_word)
            # Checking if the user is successfully authenticated
            if user is not None:
                # Login the user and creating a user session
                if user.is_active:
                    login(request, user)
                    return HttpResponseRedirect(reverse('shops:order'))
                else:
                    messages.error(request, 'User is disabled.')
            else:
                form = LoginForm()
                # Flashing a message on incorrect login credentials
                messages.error(request, 'Incorrect Login Details .. Please try again')
```

```
37      else:
38          # Creating a form instance
39          form = LoginForm()
40
41      return render(request, 'shops/login.html', {'form': form})
42
43
44  # Fetching all the orders for that user
45  @login_required(login_url='/shops/login/')
46  def user_order_view(request):
47      data_set = ShoppingTransaction.objects.filter(user=request.user)
48      return render(request, 'shops/order.html', {'orders': data_set})
49
50
51  # Order details page
52  @login_required(login_url='/shops/login/')
53  def user_cart_view(request, transaction_id):
54      data_set = TransactionDetail.objects.filter(transaction=transaction_id)
55      return render(request, 'shops/cart.html', {'items': data_set})
56
57
58  # Logout Page
59  @login_required(login_url='/shops/login/')
60  def user_logout(request):
61     logout(request)
62     return render(request, 'shops/logout.html', {})
```

In the above code snippets, which lines are vulnerable?
Select one alternative:

- views.py: 54-55 ✔
- models.py: 11-12
- models.py: 22-23
- cart.html: 10-11

19 CIA triad (1 point)

Which of the following principles is part of the CIA triad?
Select one alternative:

- Auditability: Enables monitoring and recording of system activities for security analysis.
- Accountability: refers to the principle that an individual is entrusted to safeguard and control equipment, keying material, and information.
- Attacks: involve direct actions against a system or network.
- Availability: Ensures that authorized users can access data when needed. ✔
- Authentication: Verifies the identity of users or processes.

20 Authorization vulnerability in the code (1 point)

models.py

```
from __future__ import unicode_literals

from django.db import models
from django.contrib.auth.models import User


# Model for team participating in competition.
class Team(models.Model):
    name = models.CharField(max_length=15)

    def __str__(self):
        return self.name


# Model for gamer profiles.
class GamerProfile(models.Model):
    alias_name = models.CharField(max_length=40)
    game_name = models.CharField(max_length=30)
    score = models.IntegerField()
    team = models.ForeignKey(Team)
    user = models.ForeignKey(User)

    def __str__(self):
        return self.alias_name
```

views.py

```
from django.shortcuts import render
from django.contrib.auth import authenticate, login, logout
from django.core.urlresolvers import reverse
from django.http import HttpResponseRedirect, HttpResponse
from django.contrib import messages
from django.contrib.auth import decorators
from django.shortcuts import get_object_or_404

from games.models import GamerProfile, Team
from games.forms import LoginForm


# User login process
def user_login(request):
    # Checking the request method
    if request.method == 'POST':
        # Create a form instance and populate it with data from the request
        form = LoginForm(request.POST)
        if form.is_valid():
            # Fetching the username and passwords from POST methods
            user_name = form.cleaned_data['username']
            pass_word = form.cleaned_data['password']
            # Authenticating the user
            user = authenticate(username=user_name, password=pass_word)
            # Checking if the user is successfully authenticated
            if user is not None:
                # Login the user and creating a user session
                if user.is_active:
                    login(request, user)
                    return HttpResponseRedirect(reverse('games:dashboard'))
                else:
                    messages(request, 'User is disabled.')
            else:
                form = LoginForm()
                messages.error(request, 'Incorrect Login Details. Please try ag ain')
    else:
        # Instantiating empty form
        form = LoginForm()

    return render(request, 'games/login.html', {'form': form})


# User gaming dashboard
@decorators.login_required(login_url='/games/login/')
def dashboard(request):
    team = get_object_or_404(Team, user=request.user)
    team_gamers = GamerProfile.objects.filter(team=team.team)
    return render(request, 'games/dashboard.html', {'team_gamers': team_gamers, })


# User Team members
@decorators.login_required(login_url='/games/login/')
def gamer_profile(request, gamer_id):
    gamer_details = get_object_or_404(GamerProfile, pk=gamer_id)
    return render(request, 'games/gamer_details.html', {'gamer': gamer_details, })
```

```
56
57
58 # User logout
59 @decorators.login_required(login_url='/games/login/')
60 def log_out(request):
61    logout(request)
62    return render(request, 'games/logout.html', {})
```

settings.py

```
1 # -*- coding: utf-8 -*-
2
3 #
4 # settings file for production environment
5 #
6 # This settings provides the MINIMUM level of security. Additional
7 # settings may be used to hardening the system (not added here because of
8 # potential compatibility issues with the software), like, for example:
9 #
10 #  - SECURE_PROXY_SSL_HEADER
11 #  - SECURE_HSTS_SECONDS
12 #  - SECURE_HSTS_INCLUDE_SUBDOMAINS
13 #  - SECURE_SSL_REDIRECT
14 #  - SECURE_SSL_HOST
15 #
16
17
18 from __future__ import unicode_literals
19
20 import os
21
22 from django.core.exceptions import ImproperlyConfigured
23
24 INSTALLED_APPS = [
25    'django.contrib.admin',
26    'django.contrib.auth',
27    'django.contrib.contenttypes',
28    'django.contrib.sessions',
29    'django.contrib.messages',
30    'django.contrib.staticfiles',
31    'games.apps.GamesConfig',
32 ]
33
34 ROOT_URLCONF = 'website.urls'
35
36 WSGI_APPLICATION = 'website.wsgi.application'
37
38 DEBUG = False
39
40 ALLOWED_HOSTS = [
41    'randomapp.securecodewarrior.com'
42 ]
43
44 CSRF_COOKIE_SECURE = True
45 SESSION_COOKIE_SECURE = True
46
47 try:
48    SECRET_KEY = os.environ['DJANGO__SECRET_KEY']
49
50    DATABASES = {
51        'default': {
52            'ENGINE': 'django.db.backends.postgresql',
53            'NAME': os.environ['DJANGO__DB_NAME'],
54            'USER': os.environ['DJANGO__DB_USER'],
55            'PASSWORD': os.environ['DJANGO__DB_PASSWORD'],
56            'HOST': os.environ['DJANGO__DB_HOST'],
57            'PORT': os.environ['DJANGO__DB_PORT'],
58        }
59    }
60
61 except KeyError, ex:
62    key = ex.args[0]
63    raise ImproperlyConfigured("The environment variable {0} "
64                               "was not found and is required".format(key))
65
66 MIDDLEWARE_CLASSES = [
67    'django.middleware.security.SecurityMiddleware',
68    'django.contrib.sessions.middleware.SessionMiddleware',
69    'django.middleware.common.CommonMiddleware',
70    'django.middleware.csrf.CsrfViewMiddleware',
71    'django.contrib.auth.middleware.AuthenticationMiddleware',
72    'django.contrib.auth.middleware.SessionAuthenticationMiddleware',
73    'django.contrib.messages.middleware.MessageMiddleware',
74    'django.middleware.clickjacking.XFrameOptionsMiddleware',
75 ]
76
77 TEMPLATES = [
```

```
78    {
79        'BACKEND': 'django.template.backends.django.DjangoTemplates',
80        'DIRS': [],
81        'APP_DIRS': True,
82        'OPTIONS': {
83            'context_processors': [
84                'django.template.context_processors.debug',
85                'django.template.context_processors.request',
86                'django.contrib.auth.context_processors.auth',
87                'django.contrib.messages.context_processors.messages',
88            ],
89        },
90    },
91 ]
92
93 AUTH_PASSWORD_VALIDATORS = [
94    {
95        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
96    },
97    {
98        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
99    },
100    {
101        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
102    },
103    {
104        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
105    },
106 ]
107
108 STATIC_URL = '/static/'
```

In the above code snippets, which lines of code are vulnerable?
Select one alternative:

- views.py:46-47
- settings.py:71-72
- views.py:54-54 ✔
- models.py: 20-21

21 Crypto vulnerability in the code (1 point)

```
from __future__ import unicode_literals

import hashlib
from datetime import date
from datetime import timedelta

from django.db import models
from django.conf import settings


class Payment(models.Model):
    """
    Represents a payment.

    It could be pending to be processed (`accepted` is None)
    or already processed (`accepted` is True or False).
    """
    description = models.CharField(max_length=50)
    payment_from = models.ForeignKey(settings.AUTH_USER_MODEL, related_name='+')
    payment_to = models.ForeignKey(settings.AUTH_USER_MODEL, related_name='+')
    amount = models.DecimalField(max_digits=9, decimal_places=2)
    accepted = models.NullBooleanField(null=True)


class InvalidSecurityCode(Exception):
    """The provided security code is not valid"""


class SecurityCodeManager(models.Manager):
    @staticmethod
    def encrypt_security_code(plaintext_security_code):
        """
        Encrypt the provided plain-text security code
        :param plaintext_security_code: plain-text security code
        :return: crypted security code
        """
        assert isinstance(plaintext_security_code, unicode)

        hash_inst = hashlib.md5(plaintext_security_code.encode('utf-8'))
        return hash_inst.hexdigest()

    def check_security_code(self, plaintext_security_code):
        """
        Verifies if the provided plain-text security code
        exists in the database, and isn't too old.

        Raises an exception if the code isn't valid.

        :param plaintext_security_code: the security code in plain text
                                        format (as entered by the user)
        :raise InvalidSecurityCode: if code is not valid
        """
        crypted_code = self.encrypt_security_code(plaintext_security_code)

        today = date.today()
        valid_from_date = today - timedelta(days=15)

        # control date and ignore time, we no need such precision
        qs = self.filter(created_at__date__gte=valid_from_date)
        qs = qs.filter(crypted_password=crypted_code)

        if not qs.exists():
            raise InvalidSecurityCode()

    def delete_old_security_codes(self):
        """
        Delete old security codes from the database.

        This should be called periodically to avoid having
        old codes in the database.
        """
        today = date.today()
        valid_from_date = today - timedelta(days=15)

        self.filter(created_at__date__lt=valid_from_date).delete()


class SecurityCode(models.Model):
    """
    Represents a security code to be entered by the user
    to prove the authenticity when processing a payment.
    """
```

```
83    created_at = models.DateTimeField(auto_now_add=True)
84    crypted_password = models.CharField(max_length=1000, db_index=True)
85
86    objects = SecurityCodeManager()
87
88    def set_security_code(self, plaintext_code):
89        """
90        Crypt and set the security code in this instance, based on the
91        provided plain-text security code.
92
93        Use of this method must call save() to update the database.
94        """
95
96        if len(set(list(plaintext_code.lower()))) < 10:
97            raise InvalidSecurityCode("The provided text is "
98                                      "not valid and can not be used as "
99                                      "a security code")
100
101        self.crypted_password = SecurityCodeManager.encrypt_security_code(
102            plaintext_code)
103
104    def __str__(self):
105        return "Security code {} ({})".format(self.id, self.created_at)
```

Which lines of the above code snippet are vulnerable?

Select one alternative:

- 83-84
- 72-75
- None of the above listed lines are vulnerable.
- 39-40 ✔

22 Security guiding principle (1 point)

Which security guiding principle is related to the blacklisting countermeasure?

Select one alternative:

- Promote privacy
- Keep it difficult
- Practice defense in depth
- Keep it simple
- Be reluctant to trust ✔

23 OWASP vulnerability in the code (1 point)

index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Random App</title>
</head>
<body>
{# Message notifications #}
   {% if messages %}
       <ul class="messages">
              {% for message in messages %}
                 <li {% if message.tags %} class="{{ message.tags }}"{% endif %} >{{ message }}</li>
              {% endfor %}
       </ul>
    {% endif %}
    {# Message notifications end #}
    <h4>Check host status</h4>
         <form  method="POST" action="{% url 'host:index' %}">
             {% csrf_token %}
              {{ form }}
          <br />
             <input type="submit" value="submit" />
         </form><br>
    <hr>
    <br>
    {% if request.POST %}
       <h4>{{ output }}</h4>
    {% endif%}
    <br>

</body>
</html>
```

forms.py

```
from django import forms


# Form architecture to find host status
class HostCheckForm(forms.Form):
    ip = forms.CharField()
```

views.py

```
from django.shortcuts import render

from host.forms import HostCheckForm

from os import popen2


# View method to check host status
def index(request):
    output = None
    # Checking request method
    if request.method == 'POST':
        # Initialising form with POST request
        form = HostCheckForm(request.POST)
        # Validating form inputs
        if form.is_valid():
            cmd_string = 'ping -c 3 ' + form.cleaned_data['ip']
            process_output = popen2(cmd_string, mode='r', bufsize=-1)
           output = process_output.__getitem__(1).read()
    else:
        # Initialising empty form
        form = HostCheckForm()

    return render(request, 'host/index.html', {'form': form, 'output': output})
```

In the above code snippets, which lines have vulnerability?
Select one alternative:

- views.py:16-19 ✔
- index.html:26-28
- views.py:14-14
- forms.py:5-6

24 CVSS (1 point)

What does CVSS stand for in the context of cybersecurity?
Select one alternative:

- Cyber Vulnerability Secret Service
- Critical Vulnerability Scoring System
- Common Vulnerability Security System
- Common Vulnerability Scoring System ✔
- Cybersecurity Vulnerability Severity Scale

25 Another OWASP vulnerability in the code (1 point)

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Random App</title>
    </head>

    <body>
   {# Message notifications #}
   {% if messages %}
       <ul class="messages">
              {% for message in messages %}
                 <li>{% if message.tags %} class="{{ message.tags }}"{% endif %}>{{ message }}</li>
              {% endfor %}
       </ul>
    {% endif %}
    {# Message notifications end #}
    <h4>Team Collaborator Calendar</h4>
         <form  method="POST" action="{% url 'teams:index' %}">
             {% csrf_token %}
              Email : {{form.email}}
        {{ form.email.errors }}
        Scheduled Task : {{form.event}}
        {{ form.event.errors }}
       Date : {{form.date}}
        {{ form.date.errors }}


           <br />
            <input type="submit" value="submit" />
        </form><br>
    <hr>
    <br>
    {% if calendar_events %}
       <h4>Latest events</h4>
       {% for event in calendar_events %}
       <b>Email: </b> {{ event.email }}<br>
       <b>Task:</b> {{ event.event }}<br>
       <b>Date:</b> {{ event.date }}<br><br>
       {% endfor %}
    {% endif%}

    <br>
<script src="//code.jquery.com/jquery-1.10.2.js"></script>
<script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
<script>
  $(function() {
    $( ".datepicker" ).datepicker({
      changeMonth: true,
      changeYear: true,
      yearRange: "1900:2012",

    });
  });
  </script>
    </body>
</html>
```

models.py

```
from __future__ import unicode_literals

from django.db import models


#  Model for Calendar App
class Calendar(models.Model):
    email = models.EmailField()
    date = models.DateField()
    event = models.CharField(max_length=1024)
```

views.py

```
from django.core.urlresolvers import reverse_lazy
from django.shortcuts import render
from django.utils.html import mark_safe
from django.views.generic import CreateView, TemplateView
# mark_safe tells django templates that a string should be used AS IS
from teams.forms import CalendarForm
from teams.models import Calendar


# View for scheduling task form and render scheduled task
class Index(CreateView):
```

```
12      form_class = CalendarForm
13      model = Calendar
14      template_name = 'teams/index.html'
15      success_url = reverse_lazy('teams:success')
16
17      # Custom function
18      def get_all_events(self):
19          temp_calendar_events = []
20          for events in Calendar.objects.all().order_by('-date'):
21              events.event = mark_safe(events.event)
22              temp_calendar_events.append(events)
23          return temp_calendar_events
24
25      # Method for form/POST data
26     def get_context_data(self, **kwargs):
27          context = super(Index, self).get_context_data(**kwargs)
28          final_events = self.get_all_events()
29          context['calendar_events'] = final_events
30          return context
31
32
33 # View for redirection
34 class Success(TemplateView):
35      template_name = 'teams/success.html'
```

In the above code snippets, which lines are vulnerable?
Select one alternative:

- views.py:28-28
- views.py:20-22 ✔
- index.html:22-23
- model.py:10-10

26 Public key cryptography algorithms (1 point)

Which statements regarding public key cryptography algorithm are FALSE?
1. Message sender and receiver use identical keys when they use public key cryptography algorithms.
2. The public key cryptography algorithms are usually open to public.
3. Stream cipher is a public key cryptography algorithm.
4. ECDSA is not a public key cryptography algorithm.
Select one alternative:

- 2, 3, and 4
- 1, 3, and 4 ✔
- All of them
- 1, 2, and 4

27 Buffer overflow (1 point)

Which of these kinds of inputs can cause a buffer overflow?
1. An environment variable
2. String input from the user
3. A single integer
4. A floating point number
5. File input

Select one alternative:

- All of the above ✔
- 3 and 4
- 1 and 2
- 2 and 5

28 Security requirements (1 point)

Which of these is a good security requirement?
Select one alternative:

- The system should be free from vulnerabilities
- The system must have good usability
- End user data should be encrypted at rest ✔
- The system shall work just like the previous one, but on a new platform
- The system shall encrypt all confidential data using the RSA algorithm

29 Cookies (1 point)

What is the security issue of cookie-based tokens?
Select one alternative:

- Web browser can not save the cookie value
- Web browser can not see the cookie expiration time
- Server does not see which domain sends the cookie ✔
- Cookies are unhealthy for the end user
- Server can not save the cookie value

30 Static code analysis (1 point)

Which approach does NOT belong to static code analysis for vulnerability detection?
Select one alternative:

- Control flow analysis
- Pattern matching
- Taint analysis
- Penetration testing ✔

31 Supply chain vulnerability countermeasures (1 point)

Which is NOT a transparency countermeasure of software supply chain security?
Select one alternative:

- Version Locking ✔
- Software Bill of Materials (SBOM)
- Sigstore
- Dependabot

32 Microservice architecture security countermeasures (1 point)

What is the countermeasure to defend against attacks targeting the load balancer in the microservice architecture?
Select one alternative:

- Service-level authorization
- Service-to-service authentication
- Rate throttling ✔
- Secure container

Case description: Risk assessment of a risk assessment tool for Air Traffic Management (ATM)

SESAR Joint Undertaking defines, develops and deploys technologies to transform air traffic management (ATM) in Europe. These technologies are known as *solutions* and are developed in numerous projects under det SESAR JU programme. Solutions can be used to manage conventional aircrafts, drones, air taxis and vehicles flying at higher altitudes, and need to undergo risk assessments at various stages of their development lifecycle (i.e. concept development, lab experiments, prototypes in realistic environments, proven system in operation development). One of the solutions is a cyber security risk assessment methodology, that is to be applied to the other solutions in order to derive their security requirements and maintain an up-to-date risk picture. A part of this solution is a web-based tool that is intended to make risk assessments easier for other air traffic management solutions. This includes defining scope and goals, identifying assets, threats and vulnerabilities, describing and evaluating risks and deriving security requirements. Since this web-based tool is considered to be a solution by itself, you will also need to perform a risk assessment of it (effectively a risk assessment of a risk assessment tool).

You have been given the following business goals:

- BG1: Support the ATM risk assessment methodology.
- BG2: A user-friendly web-interface that supports various stakeholders involved.
- BG3: Able to retrieve assets from a digital catalogue of reusable items (e.g. flight data, satellite datalink, primary radar, air traffic controller, passengers.).
- BG4: Preserve confidentiality of the assessments of the SESAR solutions.
- BG5: Allow sharing of risk assessment information between authorized people (involved in an assessment).

Part 1 tasks (30 points in total)

In this part you will perform tasks related web-based tool from the case description.

If you feel that any of the tasks require information that you do not find in the text, then you should:

- Document the necessary assumptions (e.g. technology, standards, software, design choices.)
- Explain why you need them.

Your answers should be brief and to the point.

Task 1: As part of defining the scope, list at least five impact dimensions you consider relevant for this assessment. (3 points)

The "correct" dimensions are:

- Personell
- Capacity
- Performance
- Economic
- Branding
- Regulatory
- Environment

Other dimensions that make sense are also accepted. Risk dimensions were explained in the Risk management lecture slides, using *Confidentiality, Availability, Financial* and *Reputation* as examples.

Task 2: What kind of people/stakeholders would you involve in the assessment? Explain why. (3 points)

The threat modelling lecture gave examples of people to involve, such as

- Security experts
- Developers
- Former hackers (not really recommended)

The important reason is the pattern from the Threat modelling manifesto:

"Assemble a diverse team with appropriate subject matter experts and cross-functional collaboration." This ensures different viewpoints.

In this case, it would be natural to also involve e.g. air traffic controllers, Eurocontrol/regulators, pilots, system administrators and so on. At least three relevant stakeholders give full score.

Task 3: What kind of access control model would you recommend for this solution? (2 points)

The best solution here would be Mandatory Access Control since there is a need to preserve the confidentiality of the assessment in a controlled manner (centraliazed). A single point can be given for other good suggestions.

Task 4: Identify 5 assets (something of value that needs protection) related to the tool. (3 points)

Here we accept both primary assets (information, services) and supporting assets (physical). These could for instance be:

Risk assessment result

Asset catalogue

Passengers

Aircrafts

Risk assessment software

Network

User credentials

Other sensible assets are also accepted. Less than 5 reduces points.

Task 5: In the use case diagram in the next page, you can see use cases and undefined actors. Define at least 5 suitable actors and describe how they should be connected referring to the use case labels (you can add more actors if needed). (3 points)

See example solution. The actors could be similar to the ones in task 2. Less than 5 reduces points.

Task 6: Define at least 5 corresponding misuse case elements and describe how they should be connected to the use cases (you use the labels as a references). (3 points)

See example solution. Less than 5 reduces points.

Task 7: The last page shows a DFD. Explain the different types of elements in the diagram. Identify possible attack points in relation to the DFD elements and describe at least 5 threats according to STRIDE. (4 points)

Here there are many possible solutions. The important think is sensible threats and that they have been categorized according to STRIDE (like they did in exercise 4). Example:

Attack point	**STRIDE category**	**Threat**
User	Spoofing	Steal access credentials
User	Denial of Service	Food poisoning (actually in the real threat catalogue of SecRAM)
Risk assessment tool	Denial of Service	DDoS attack against web server
Risk assessment tool	Information disclosure	Usage rights changed so that a user can access assessments for other solutions.
Model store	Tampering	Models deleted by unauthorized user.

Asset catalogue	Tampering	Assets changed so that solutions loose reference assets.
Administrator	Spoofing	Steal access credentials
User database	Tampering	Access rights changed

Task 8: Based on the threat models, identify at least 5 technical risks and evaluate them. You should describe necessary assumption related to the technology. (3 points)

Here, the candidate can take threats already identified in the misuse case or DFD diagrams. The important thing is a justified evaluation, using either system-centric and/or attacker-centric approaches.

Note that the in the risk management lecture we explicitly state that determining impact of technical risks can be difficult and therefore not recommended by Gary McGraw. Full points can be given without impact assessment. Risk can also be ranked. Less than 5 technical risks reduces points.

Examples:

Threat; Likelihood; impact

TR1: Service disruption of assets catalogue; Low (easy attack since there is a single point of failure, cheap, but little motivation for most attackers); Low (a risk assessment tool is not time critical): Overall risk: Low.

TR2: Steal access credentials by threatening administrator; Medium (Needs highly motivated attacker, access to models can be used to plan attack against solutions); High (Models are considered confidential, loss of confidentiality); Overall risk: High

TR3: Modify risks of the risk assessment; Low (A competitor might want to sabotage the acceptance of a solution, likely to be detected by the solution developers); Medium (an unsafe solution is accepted into operations); Overall risk: Medium

Task 9: Select at least 3 technical risks and define one well-formulated security requirement for each. (3 points)

For full score the security requirements should be according to recommendations by Firesmith (e.g., should be proper sentences, what the system should do (not what should not be done), not two in one, allow for different solutions, testable). Less than 3 reduces points.

For instance:

TR1: Service disruption of assets catalogue:

SR1: The system should provide redundancy of the asset catalogue server.

TR2: Steal access credentials by threatening administrator:

SR2: The system should require 2FA for all users.

TR3: Modify risks of the risk assessment

SR3: All changes to the risk values should be reviewed by at least two certified users.

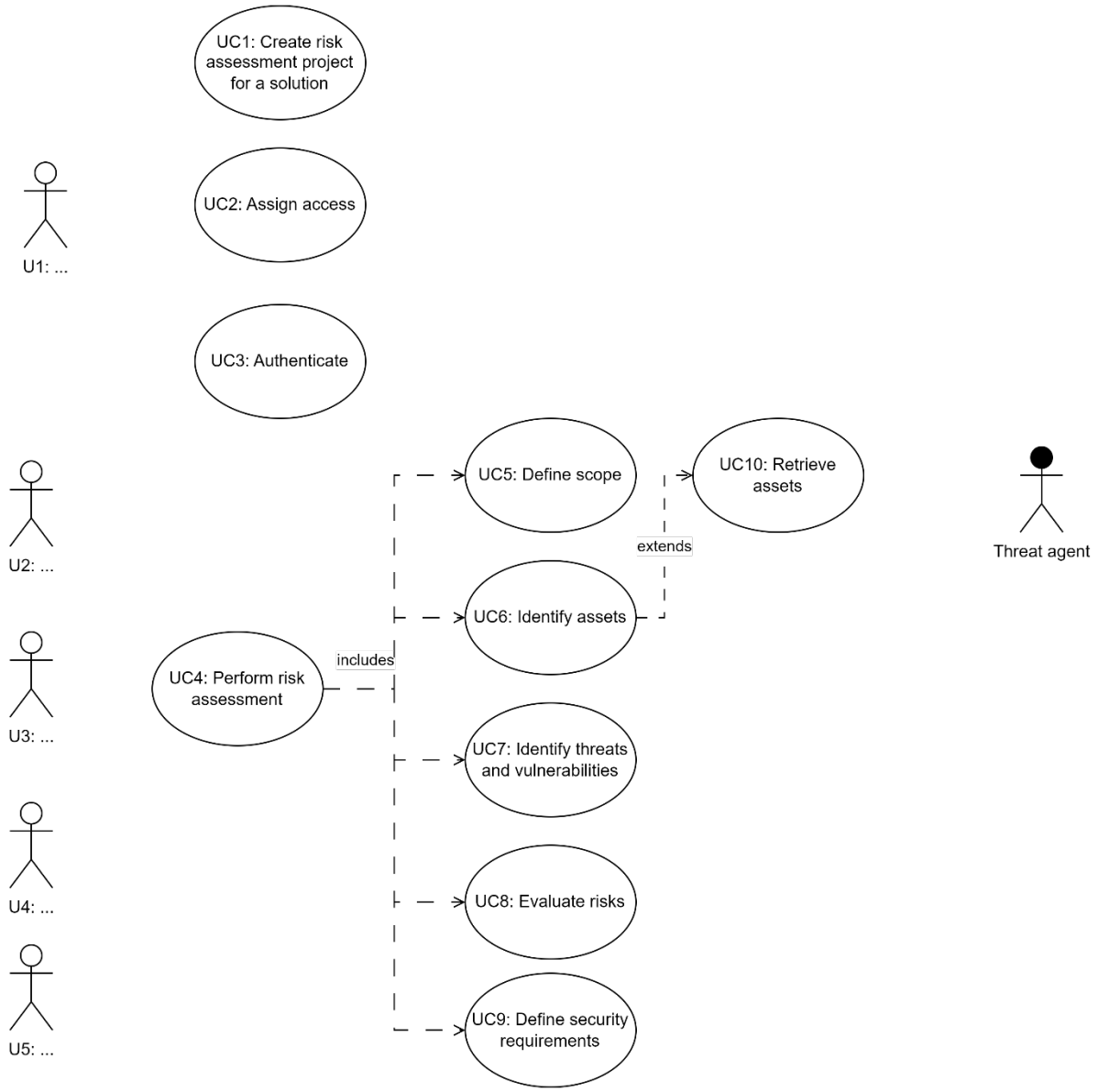
Task 10: Write a short reflection on which threat agents you consider to be significant for this tool. Justify these using principles from attacker-centric threat modelling. (3 points)

The candidate might have covered this partly in task 8 when evaluating risks. Reference to previous answers is allowed, but here we expect a somewhat deeper reflection. One threat agent is enough if the reflection is solid.

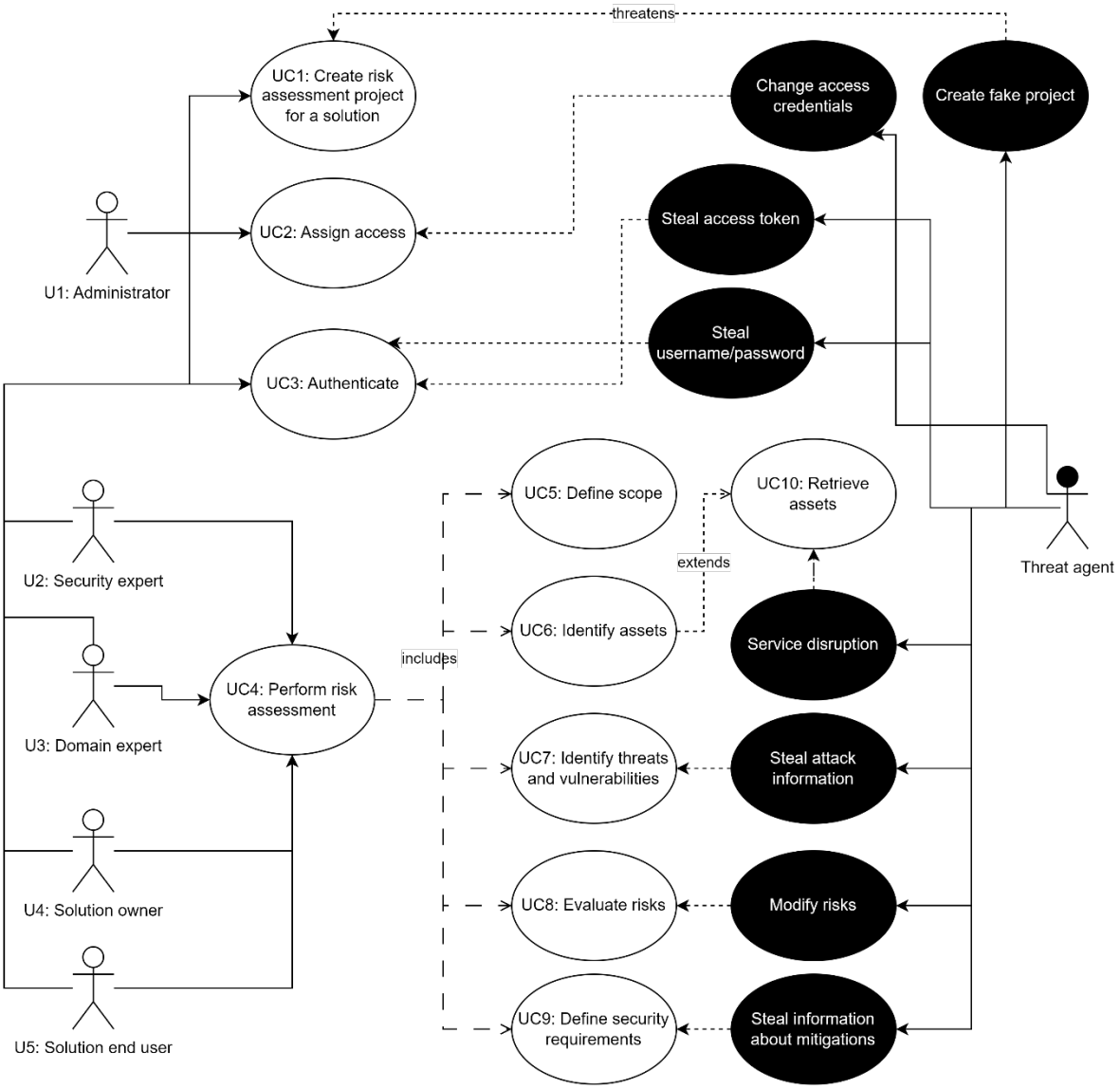
Threat agents that want to sabotage aviation solution might want to cause fear, uncertainty and doubt among passengers/citizens. State actors/government cyber warriors with good access to resources are probably the most likely agents.

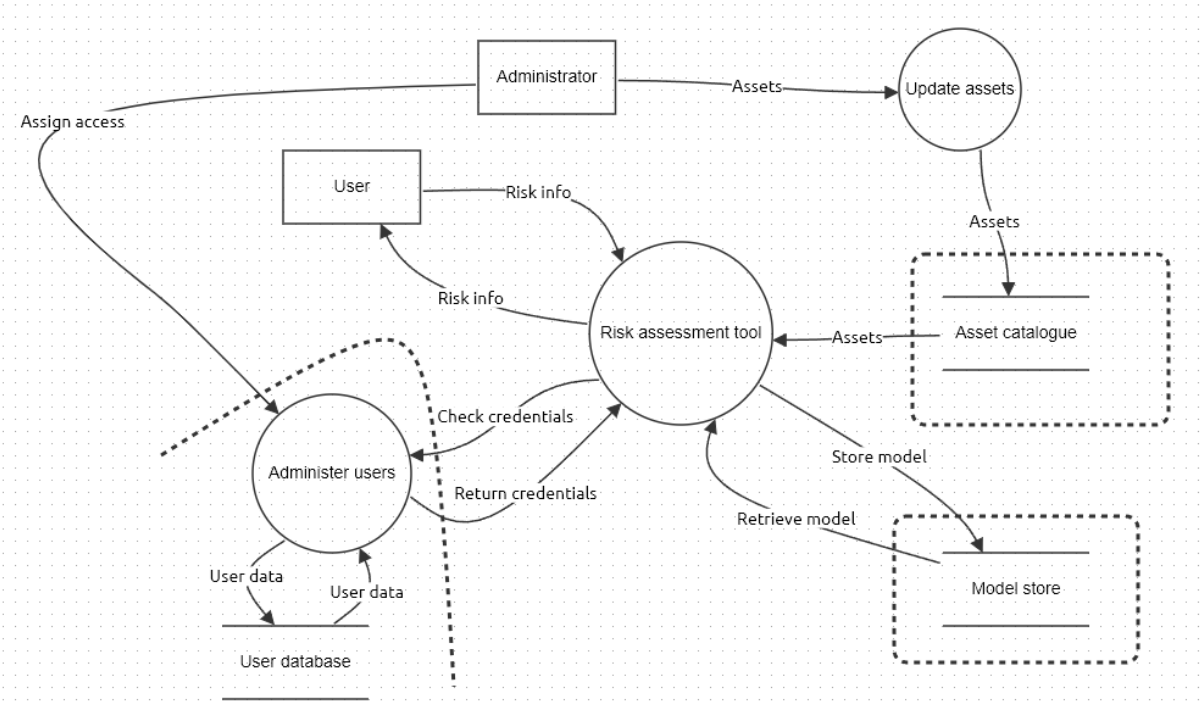
Another scenario might be that competitors want to sabotage the development of a technology that could put them out of business. Aviation is a high-cost industry.

Insiders might want to sell information to highest bidder or demand ransom. Security clearance should be in place, but there are many users of the risk assessment tool.


UC1: Create risk assessment project for a solution
UC2: Assign access
UC3: Authenticate
U1: ...
U2: ...
U3: ...
U4: ...
U5: ...
UC4: Perform risk assessment
includes
UC5: Define scope
UC6: Identify assets
UC7: Identify threats and vulnerabilities
UC8: Evaluate risks
UC9: Define security requirements
extends
UC10: Retrieve assets
Threat agent

Example solution:




Administrator
Assets
Update assets
Assign access
User
Risk info
Assets
Risk info
Risk assessment tool
Assets
Asset catalogue
Check credentials
Administer users
Store model
Return credentials
Retrieve model
User data
User data
Model store
User database