Development Team Project: Design Document

Domain:
Dutch Police Internet Forensics

Marios Maragkos
&
James Hines

Word Count: 596

Table of Contents

Word Count: 596

Description

In the era of computers, smartphones, and other data-collecting devices, digital evidence has become more significant in solving crimes and other legal matters, especially with the increasing usage of computers and data-collecting devices in everyday life. Computer forensics involves data recovery with legal compliance guidelines to make the information admissible in court (Lutkevich, n.d). The current report aims to provide the Dutch forensics police department with the necessary system and software tools to fulfil its tasks and responsibilities legally.

Functional Requirements

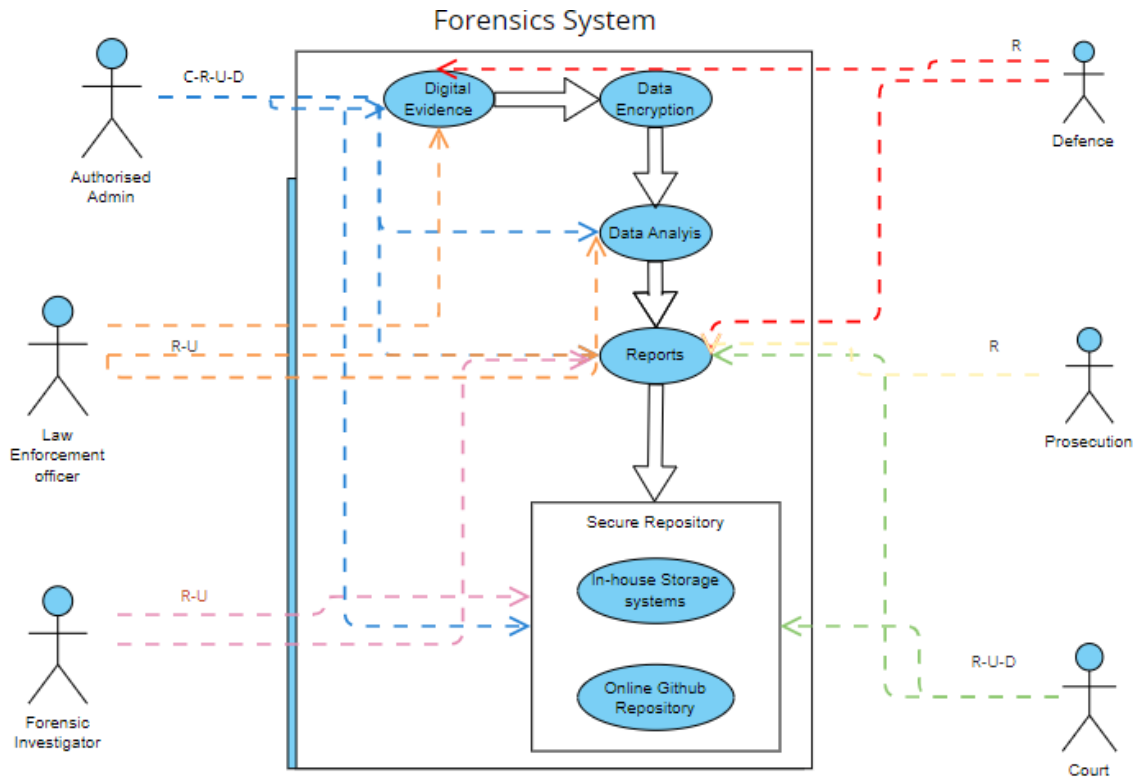

Figure.1 Use Case Diagram

Problem Definition

Figure.1 illustrates the CRUD attitudes of the secure software Create, Read, Update and Delete operations in diverse data types incorporated in different roles using the Authentication, Authorisation and Accounting AAA security framework (Escott, 2020). Example, the actor identified (Authentication) because the court has fewer rights on data than the authorised (Authorisation grants and restricts rights) admin. In addition, Confidentiality/Integrity/Availability triad ensures the development phase of data classification systems and access privileges management is easier (Fortinet, n.d).

Software and Hardware Resources

Software Resources	Hardware Resources
• Python3	• Laptops
• Ubuntu	• Ubuntu Linux workstations
• Open source repositories, in this case, GitHub	• Mobile Devices
• Network analysis tools such as Wireshark (Poston, 2021)	

Table.1 Software and Hardware Resources (Hillenius, 2013)

Libraries

Libraries	Function
• **Passlib**	Password hashing library for Python
• **Hashlib**	Secure hashes and message digests
• **Getpass**	Prompt the user for a password without echoing
• **Sqlite3**	Disk-based database that doesn't require a separate server process and allows prototyping before porting code
• **Scapy**	Network analysis, penetration testing, and forensic investigation tools widely used in cybersecurity.
• **PyCrypto**	Implements cryptographic functions such as encryption, decryption, digital signatures, and hash functions
• **Requests**	Requests is a well-known Python package for sending HTTP requests and handling the responses
• **BeautifulSoup**	BeautifulSoup is a popular Python module for web scraping and HTML and XML document processing
• **Paramiko**	The Python package Paramiko implements the SSH protocol for secure remote access to servers and other network devices
• **Scikit-learn**	Scikit-learn is a prominent open-source Python machine-learning package that offers a variety of tools for data analysis, data mining, and machine learning
• **Tornado**	Tornado is a Python web framework and asynchronous networking toolkit for creating scalable, high-performance online applications
• **Nmap**	Nmap is a free, open-source network exploration and security auditing software (Geeksforgeeks, 2023).

Table.2 Bulleted list of Libraries and their function

Non-functional requirements

Threat reduction coding in Python

According to Pillai (2017:317), the following table highlights strategies for ensuring secure coding practices in python. This is specific to the Python language and is an aid to support the STRIDE threat model.

Threats in python code	Mitigations in python code
Overflow: Non-Pure buffer overflow errors	Catch 'TypeError' or 'overflow' exceptions
Files: Open file handles	Instead, Close file descriptor
Passwords: comparing original data in memory	Instead, Compare cryptographic hashes
Local Data: local stack exploit can access all data	Store sensitive data or hashed modules separately
Evaluating expressions: eval and/or exec	Point away from user input strings, APIs, data read libraries.
String Formatting: %s interpolation	Instead, Use template strings

Table.3 Securing code in Python (Pillai, 2017:317)

STRIDE

An application threat model, such as STRIDE, assesses threats during software development. It is easier and cheaper to address potential weaknesses early in the software development lifecycle when they are cheaper, easier to mitigate, and, if necessary, easier to fix if discovered.

- **Spoofing** - Falsifies identities to gain access to critical user data by impersonating a trusted source. Eg: cookie replay, session hijacking, and cross-site request forgery (CSRF) attacks.
- **Tampering** - Modifying Data/Code of an application can compromise the Confidentiality/Integrity/Availability tenet. Eg: Cross-site scripting, code injection attacks.
- **Repudiation** - To perform prohibited operations without being able to trace them.
- **Information Disclosure** - May happen intentionally or accidentally due to mistakes during code development.
- **Denial of Service** - Most common example is a buffer overflow attack which sends too much traffic to the application, eventually making it unavailable.
- **Elevation of Privilege** - Unauthorised Privileged access to critical information (Cynance, n.d)

	Threat	Violated Property	Mitigation Counter Measures
S	Spoofing Identity	Authentication	1. Encryption utilization to safeguard credentials and authentication tokens while they are being stored and transported 2. Protocols resistance to dictionary, replay, and brute force attacks 3. Strict password regulations 4. SQL authentication is substituted with trusted server authentication. 5. Salted hashes are employed to store passwords 6. Resetting passwords hides usernames and password hints (Conklin, n.d).
T	Tampering with sensitive data	Integrity	**Input Validation:** 1. Mandatory inspections for data types, formats, length, and range. 2. The client validates all data sent by him/her. 3. No security decision must be relied on variables that can be changed, such URL parameters 4. Utilization of allow list validation for input filtering 5. Implementation of Content Security Policy 6. Use of output encoding (Conklin, n.d).
R	Repudiation	Non-Repudiation	**Auditing and Logging:** 1. Passwords and other sensitive data should not be logged 2. Log files are subject to access controls (like ACLs) to prevent unauthorized access 3. In order to enable non-repudiation, integrity rules (such as signatures) are enforced on log files 4. Log files allow for the logging of important events and an audit trail for sensitive processes 5. Several servers throughout the tiers have auditing and logging enabled (Conklin, n.d).
I	Information Disclosure	Confidentiality	1. Only certified symmetric block ciphers and key lengths should be used. AES-128, AES-192, AES-256 and 3DES. 2. Use of approved MAC/HMAC/keyed hash algorithms 3. Addition of digital signature to critical database securable 4. Use of only approved cryptographic hash functions (SHA256, SHA384, SHA512) 5. Storage of Cryptographic Keys securely on IoT Device 6. Use of strong encryption algorithms to encrypt data in the database (Microsoft, 2022).
D	Denial Of Service	Availability	1. Deployment of Web Application Firewall on premised and on cloud 2. Use of Content Delivery Network (CDN) 3. Utilization of Elastic Load Balancer for smart distribution of incoming application traffic (Amazon, n.d).
E	Elevation of Privilege	Authorization	1. Verification that appropriate ACLs are set up to prevent unauthorized access to resources 2. User-specific application content that is critical is kept in the user-profile directory. 3. Deployed programs are executed with the fewest possible privileges. Resources and material cannot be forced to be accessed or enumerated. 4.Firewall Configuration to restrict access 5.Removal of unnecessary access from user roles 6.Closure of unused network ports (Microsoft, 2022)

Table.4 Recommended *STRIDE Threat and Mitigation measures*

Quality attributes

- See Table 5 below

Design Requirements

Table.5 illustrates quality attributes of the design, this provides mitigations against network congestion to allow scalability (Pallai, 2017:195).

Choice	Concurrency	Latency	Performance	Scalability
X	**High**	**Low**	**High**	**High**
	High	High	Variable	Variable
	Low	High	Poor	Poor

Table.5 Optimising for Scalability and network congestion reduction

High-level Solution Design

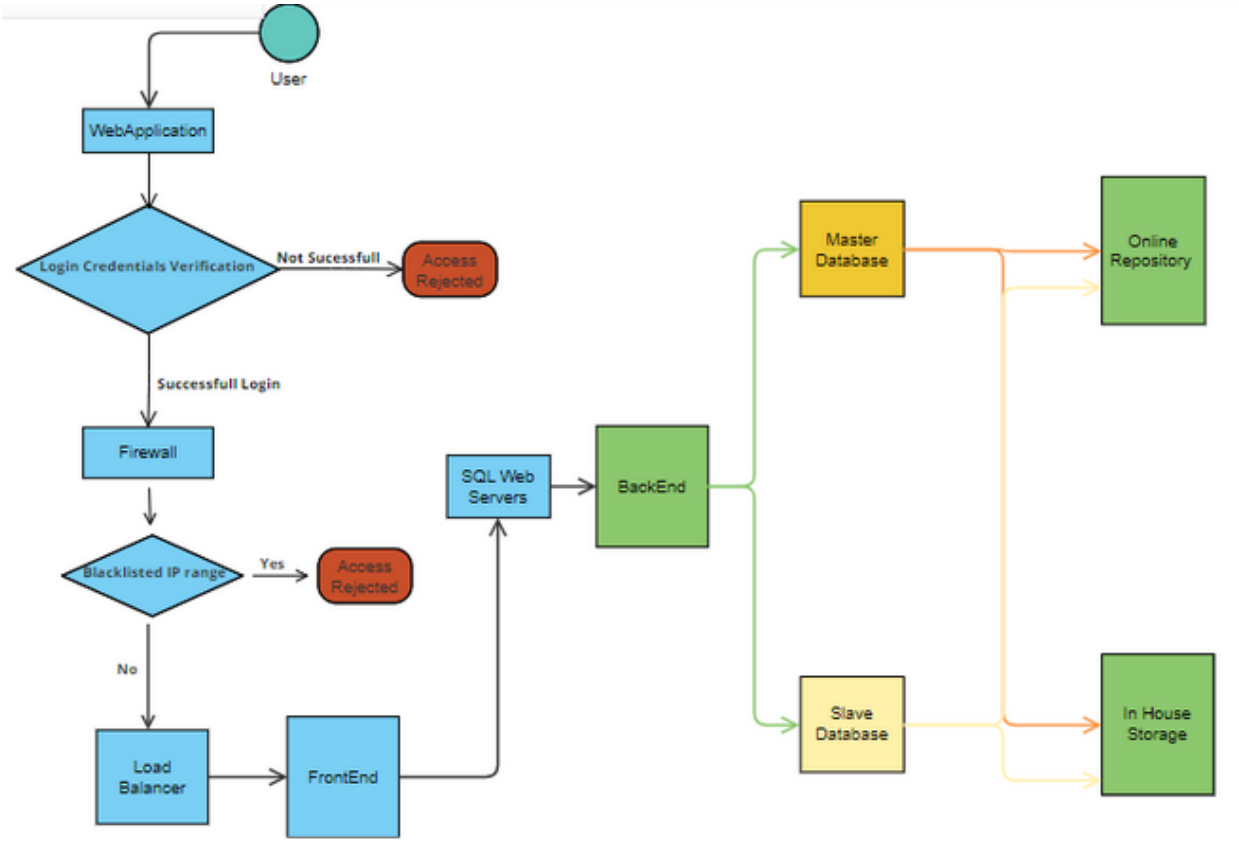

Figure 2. Activity Diagram System Architecture

Assumptions

- Vetted Dutch police internet forensic staff will be using the system
- The hosting platform supplied by the National Cyber Security Centre
- Current systems will remain in place until the new system has been created, tested and fully deployed.

Project management

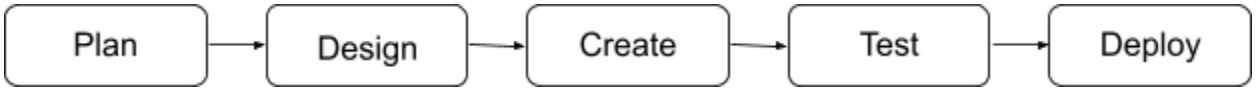

Figure.3 Software Design Lifecycle

Approach and schedule

- Spiral Waterfall approach
- Each step must be secure and complete before moving on (Figure.2)

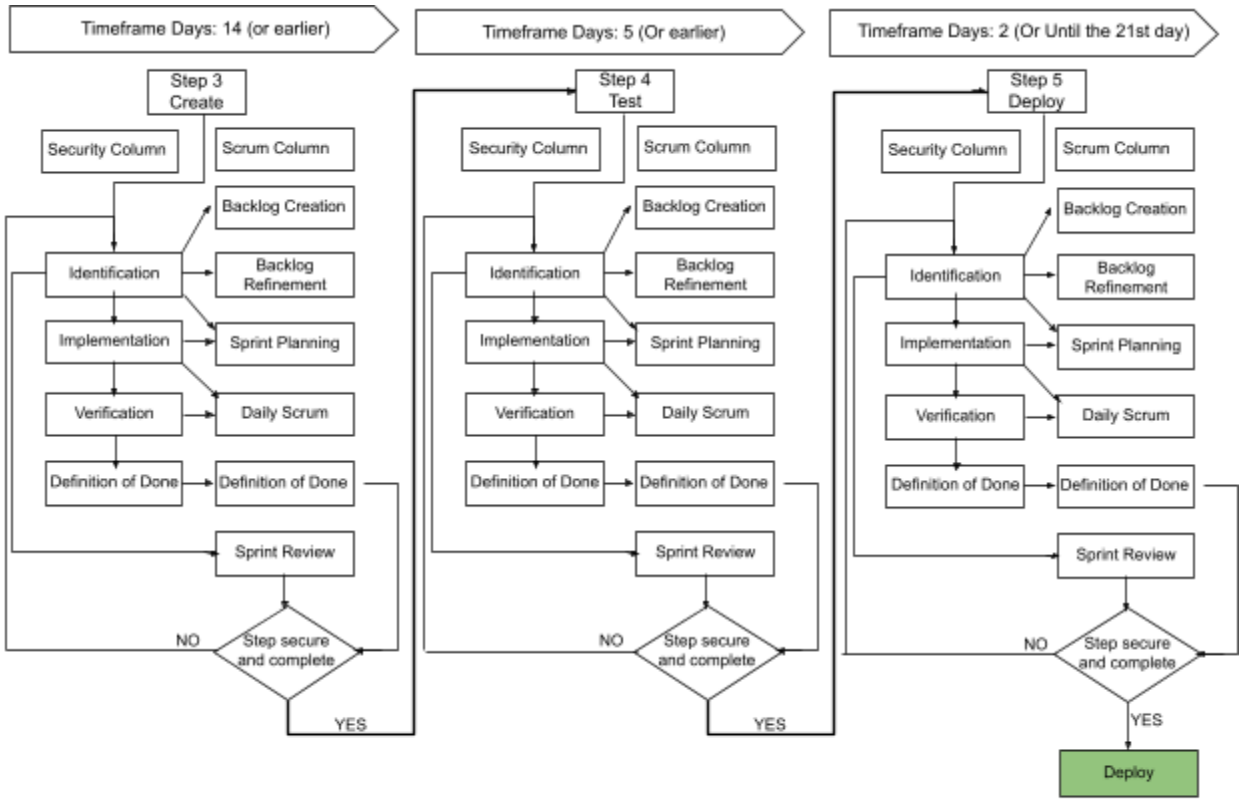

Figure 4: Security and scrum SDL (adapted from Pohl & Hof, 2015)

Compliance and privacy

- EU GDPR
- Dutch GDPR Implemented Act (Uitvoeringswet Algemene Verordening Gegevensbescherming)

According to GDPR (2022), Personal data may not be processed unless there is at least one legal basis to do so. Dutch law states that “Article 41(1) UAVG allows controllers to disregard the data subjects’ rights (Art. 12- 21 GDPR) and the obligation of breach communication to the data subject (Art. 34 GDPR)” (activeMind.legal, 2023; Onetrust, 2023).
Particular ICO & UK GDPR rules may be superseded by UAVG laws (activeMind.legal, 2023; Onetrust, 2023)

References

Amazon (n.d) What is a DDoS Attack?
Available from:https://aws.amazon.com/shield/ddos-attack-protection/ [Accessed 25 March 2023]

Conklin, K.(n.d) Threat Modeling Process.
Available from:https://owasp.org/www-community/Threat_Modeling_Process#stride-threat-list [Accessed 25 March 2023]

Cynance (n.d) STRIDE Threat Modelling: Six Steps to a Secure Application
Available from:https://www.cynance.co/stride-threat-modelling-6-steps-to-secure-apps/ [Accessed 22 March 2023]

Data subjects' rights under Dutch data protection law
https://www.activemind.legal/law/nl-data-subjects-rights/ [Accessed on 24th March 2023]

Escott, E. (2020) How do you secure your data using CRUD?
Available from:https://codebots.com/crud/how-do-you-secure-your-data-using-crud [Accessed 26 March 2023]

Fortinet (n.d) CIA triad
Available from:https://www.fortinet.com/resources/cyberglossary/cia-triad [Accessed 26 March 2023]

GDPR. (2022) Complete guide to GDPR compliance. Available from: https://gdpr.eu/ [Accessed on 24th March 2023]

Geeksforgeeks (2023) Top 10 Python Libraries for Cybersecurity.
Available from:https://www.geeksforgeeks.org/top-10-python-libraries-for-cybersecurity/ [Accessed 26 March 2023]

Hillenius, G. (2013) 'Open source only' at Dutch police Internet forensics. Available from:https://joinup.ec.europa.eu/collection/open-source-observatory-osor/news/open-source-only-dutch-p [Accessed 26 March 2023]

Lutkevich, B. (n.d) computer forensics (cyber forensics).
Available from:https://www.techtarget.com/searchsecurity/definition/computer-forensics [Accessed 26 March 2023]

Microsoft (2022) Security Frame: Authorization | Mitigations.
Available from:https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-authorization [Accessed 25 March 2023]

Microsoft (2022) Security Frame: Cryptography | Mitigation.
Available from:https://learn.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-cryptography [Accessed 25 March 2023]

OneTrust, (2023) Netherlands - Data Protection Overview. *oneTrust DataGuidance.*Available from: https://www.dataguidance.com/notes/netherlands-data-protection-overview [Accessed on 24th March 2023]

Pillai, A.B. (2017) *Software architecture with Python*. Packt Publishing Ltd. Available from: [Accessed on 23th March 2023]

Pohl, C. & Hof, H, J. (2015) Secure scrum: Development of secure software with scrum. *The Ninth International Conference on Emerging Security Information, Systems and Technologies*. Available from: https://arxiv.org/abs/1507.02992 [Accessed on: 18th March 2023]

Poston, H. (2021) Popular computer forensics top 19 tools [updated 2021].
Available from:https://resources.infosecinstitute.com/topic/computer-forensics-tools/ [Accessed 26 March 2023]