A Design Proposal for a New Secure Repository for the Dutch Police Internet Forensics

Secure Software Development

Introduction

This design document discusses the system requirements, security challenges and design of a new secure repository for the Dutch Police Internet Forensics organisation.

Web Application Functionality

- Provides latest cybersecurity news, alerts, allowing subscriptions.
- Allows organisations to register, report, and manage security incidents.
- Allows individuals to register, report and manage GDPR breaches.

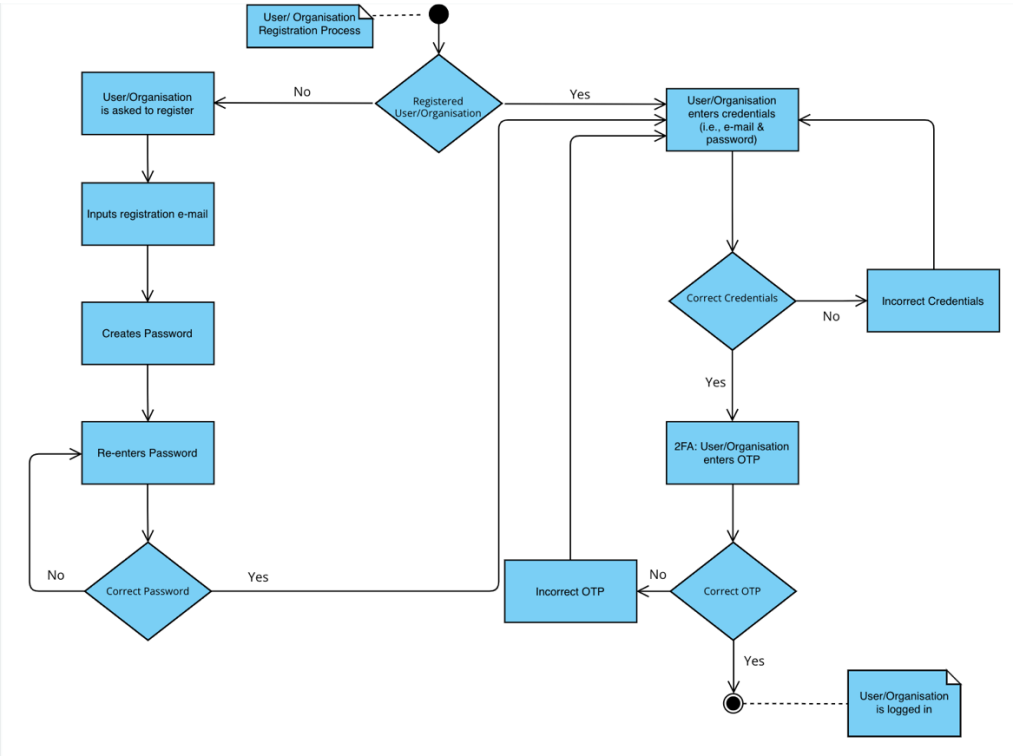

Figure 1: UML activity diagram demonstrating user/organisation registration/login process.

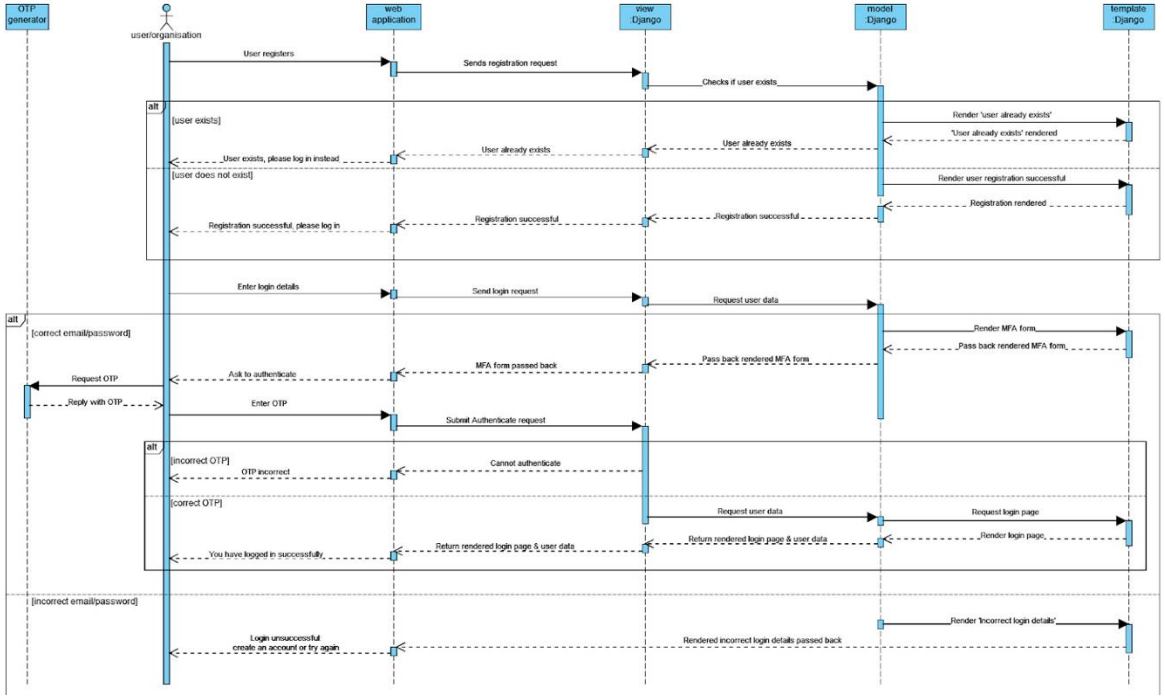

Figure 2: UML sequence diagram representing user/organisation login/registration process using Django MVT architecture.

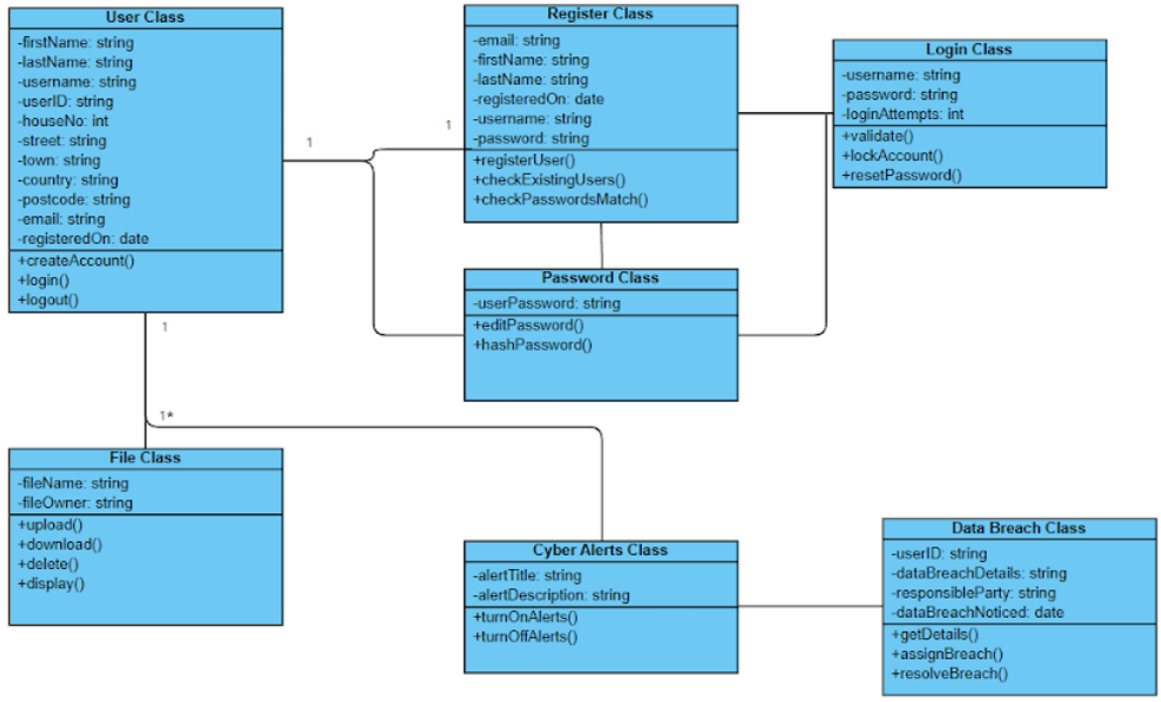

Figure 3: UML class diagram demonstrating main classes and attributes.

System Requirements & Assumptions

To meet the needs of the Dutch Police Internet Forensics, the system design should incorporate the following:

- System will act as an evidence repository for primarily unstructured data related to forensic investigations (Kebande et. al., 2021).
- As of 2013, the organisation had 9,500 staff (Hillenius, Gijs, 2013). The system may additionally need to be accessed by other law enforcement organisations.
- The system must meet all applicable privacy and security regulations (e.g., GDPR).
- The system should implement Forensics as a Service (FaaS) to take advantage of cloud-based, elastic CPU and storage capabilities (Nanda & Hansen, 2016).

Design Decisions, Tools, and Libraries

- Python 3.10 and Django 4.0.5 open-source framework (Ghimire, 2020) will be used.
- Djago's serverless SQL database SQLite is to be utilised (Tiwari et. al., 2019). See Appendix A.
- Passwords will be hashed and salted using SHA512 and the Python Secrets module for generating a secure random salt (Al Farawn et. al., 2019). The hash will be iterated 100 times to increase unhashing time (Ji et. al., 2017). Database to store salted hash passwords.

- Traffic between database and web application will be communicated over HTTPS (Kamanin, 2021).
- The Django-cryptography symmetric encryption package (PyPI, 2022) will be used to encrypt sensitive user data.
- User interface to be developed with HTML and CSS stylesheets (Tiwari et. al., 2019).
- Code will be improved with Python linters (Pyling, Pyflakes) (Farah, 2014).

Security Challenges

Relevant vulnerability identification and mitigation are based on OWASP's Top 10 (2021a) framework.

Table 1: Security vulnerabilities and mitigations.

OWASP Category	Vulnerability Mitigation
A01:2021-Broken Access Control	Install Django session framework, enable user authentication tools (Holovaty & Kaplan-Moss, 2009).
A02:2021-Cryptographic Failures	Encrypt sensitive data, proper key management (Mattsson, 2005).
A03:2021-Injection	Validate user input, use regex patterns (Aborujilah, 2022). Restrict field length to only what is necessary (Holovaty & Kaplan-Moss, 2009).
A04:2021-Insecure Design	Website URLs won't pass sensitive user data, e.g., user ID (Holovaty & Kaplan-Moss, 2009).

A05:2021-Security Misconfiguration	Build a minimalist platform (no unnecessary features, components, samples, documentation) (OWASP, 2021c).
A06:2021-Vulnerable and Outdated Components	Use dependency checkers to check for outdated/vulnerable components (Maier et. al., 2019).
A07:2021-Identification and Authentication Failures	Implement Multi-Factor Authentication (Ometov et.al., 2018).
A08:2021-Software and Data Integrity Failures	Verify software/data sources via digital signatures (OWASP, 2021e).
A09:2021-Security Logging and Monitoring Failures	Log login, access control, and server-side input validation failures with adequate user context and hold for enough time to allow forensic analysis (OWASP, 2021d).
A10:2021-Server-Side Request Forgery (SSRF)	Do not send raw responses to clients and enable authentication on all services (Zlojic, 2022).

Architectural and design patterns, SDLC approaches

- The Django MVT pattern to be used for the request routing separation from the user interface (Holovaty & Kaplan-Moss, 2009).

- Object oriented paradigm will be implemented to construct code elements (Rumbaugh et. al., 2003).
- A secure Scrum SDLC will be used:
    - Requirements/Design phases to include security requirements
    - Implementation stage backlog to include security issues
    - Verification phase to cover security testing (Maier et. al., 2017).

Conclusion

This design document includes system requirements and assumptions for a prototype of the Dutch Police Internet Forensics web application. Security challenges alongside mitigation techniques are identified and highlighted in Table 1. The web application will use these techniques to mitigate its potential vulnerabilities. Design decisions are related to the production of the prototype and certain aspects of the design must be revisited before releasing the application to the public.

References

Aborujilah, A., Adamu, J., Shariff, S.M. and Awang Long, Z. (2022). *Descriptive Analysis of Built-in Security Features in Web Development Frameworks*. [online] IEEE Xplore.

Al Farawn, A., Salih Ali, N. & Alattar, M. (2019) Anti-Continuous Collisions User Based Unpredictable Iterative Password Salted Hash Encryption. *International Journal of Internet Technology and Secured Transactions*, 1(1): 1.

Farah, G., Tejada, J.S. and Correal, D. (2014). OpenHub: a scalable architecture for the analysis of software quality attributes. *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*.

Ghimire, D. (2020) Comparative study on Python web frameworks: Flask and Django. Available from: https://www.theseus.fi/handle/10024/339796 [Accessed 20 June 2021].

Hillenius, Gijs (2013). 'Open source only' at Dutch p…. [online] Joinup. Available at: https://joinup.ec.europa.eu/collection/open-source-observatory-osor/news/open-source-only-dutch-p [Accessed 27 Jun. 2022].

Holovaty, A. & Kaplan-Moss, J. (2009). *The definitive guide to Django web development done right ; [Django is a framework that saves you time and makes Web development a joy ; updated for Django 1.1]*. Berkeley, Calif. Apress [U.A.].

Ji, R., Liu, H., Cao, L., Liu, D., Wu, Y. & Huang, F. (2017). Toward Optimal Manifold Hashing via Discrete Locally Linear Embedding. *IEEE Transactions on Image Processing*, [online] 26(11): 5411–5420.

Kamanin, T. (2021) How to run a local Django development server over HTTPS with a trusted self-signed SSL certificate. Available from: https://timonweb.com/django/https-django-development-server-ssl-certificate/ [Accessed 20 June 2021].

Kebande, V.R., Karie, N.M., Kim-Kwang Raymond Choo and Sadi Alawadi (2021). Digital forensic readiness intelligence crime repository. [online] ResearchGate. Available at: https://www.researchgate.net/publication/349367312_Digital_forensic_readiness_intelligence_crime_repository [Accessed 27 Jun. 2022].

Maier, P., Ma, Z. & Bloem, R. (2017) 'Towards a Secure SCRUM Process for Agile Web Application Development', *ARES '17: International Conference on Availability, Reliability, and Security.* Reggio Calabria, Italy, 29 August - 1 September. New York: Association for Computing Machinery. 1-8.

Mattsson, U. T. (2005). Database Encryption - How to Balance Security with Performance. *SSRN Electronic Journal* 1(1): 1-16.

Nanda, S. and Hansen, R.A. (2016). Forensics as a Service: Three-Tier Architecture for Cloud Based Forensic Analysis. [online] ResearchGate. Available at: https://www.researchgate.net/publication/301553168_Forensics_as_a_Service_Three-Tier_Architecture_for_Cloud_Based_Forensic_Analysis [Accessed 27 Jun. 2022].

OWASP (2021a) Top 10:2021 List. Available from: https://owasp.org/Top10/ [Accessed 23 June 2022].

OWASP (2021b) A05:2021 - Security Misconfiguration. Available from: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/ [Accessed 25 June 2022].

OWASP (2021c) A08: Software and Data Integrity Failures. Available from: https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/ [Accessed 25 June 2022].

OWASP (2021d) A09:2021 - Security Logging and Monitoring Failures. Available from: https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/ [Accessed 25 June 2022].

Petsas, T., Tsirantonakis, G., Athanasopoulos, E. & Ioannidis, S. (2015) 'Two-factor Authentication: Is the World Ready? Quantifying 2FA Adoption', *EuroSys '15: Tenth EuroSys Conference 2015.* Bordeaux, France, 21 April. New York: Association for Computing Machinery. 1-7.

PyPI (2022) django-cryptography 1.1. Available from: https://pypi.org/project/django-cryptography/ [Accessed 24 June 2022].

Ometov, A., Bezzateev, S., Mäkitalo, N., Andreev, S., Mikkonen, T. & Koucheryavy, Y. (2018). Multi-Factor Authentication: A Survey. *Cryptography* 2(1): 1-31.

Rumbaugh, J. Ralston, A., & Reilly, E. (2003) *Object Oriented Analysis and Design (OOAD) The Encyclopaedia of Computer Science.* 4th Ed. UK: John Wiley and Sons Ltd.

Tiwari, U., Mehfuz, S., Sharma, S. & Pandey, V.T. (2019). *Design of Python Based Lost and Found Website for College Campus*. [online] IEEE Xplore.

Zlojic, A. (2022) Server Side Request Forgery (SSRF) Attacks and How to Prevent Them. Available from: https://brightsec.com/blog/ssrf-server-side-request-forgery/ [Accessed 28 June 2022].

Appendix

Appendix A - Data dictionary

Application database

	Field	Type	Null	Default
organisation	orgName	varchar(60)	No	
	orgHouseNo	varchar(4)	No	
	orgStreet	varchar(100)	No	
	orgTown	varchar(60)	No	
	orgCountry	varchar(60)	No	
	orgPostCode	varchar(9)	No	
	orgKvkNo	varchar(8)	Yes	NULL
	orgSectorId	int(3)	Yes	-1
	orgEmail	varchar(254)	No	
	orgPhoneNo	int(15)	Yes	NULL
	registeredOn	timestamp	No	

organisationSector	sectorName	varchar(30)	No	
person	firstName	varchar(35)	No	
	lastName	varchar(35)	No	
	houseNo	varchar(4)	No	
	street	varchar(100)	No	
	town	varchar(60)	No	
	country	varchar(60)	No	
	postCode	varchar(9)	No	
	email	varchar(254)	No	
	phoneNo	int(15)	Yes	NULL
	registeredOn	timestamp	No	
cyberAdvice	adviceDescription	varchar(256)	No	
	adviceContent	text(max)	No	
organisationIncident	orgId	int(11)	No	
	incidentSummary	varchar(1000)	No	
	requiresFollowUp	int(1)	No	0
	internalIncidentId	varchar(30)	Yes	NULL
	internalInvestigation	varchar(1000)	No	

	impactId	int(11)	No	
	impactDescription	varchar(1000)	No	
	currentStateId	int(11)	No	
	othersNotified	varchar(1000)	No	
	dateOfIncident	date	No	
	timeOfIncident	time	No	
	createdDate	timestamp	No	
gdprDataBreach	personId	int(11)	No	
	dataBreachDetails	varchar(1000)	No	
	responsibleParty	varchar(1000)	No	
	othersNotified	int(1)	No	
	dateBreachNoticed	date	No	
impact	impact	varchar(30)	No	
currentState	currentState	varchar(30)	No	
newsletter	userId	int(11)	Yes	NULL
	email	varchar(254)	Yes	NULL
	subscribedOn	timestamp	No	
	unsubscribedOn	int(1)	Yes	NULL

cyberAlert	alertTitle	varchar(60)	No	
	alertDescription	text	No	

Hashed passwords database

hashedPerson	personId	int(11)	No	
	password	varchar(128)	No	
hashedOrganisation	organisationId	int(11)	No	
	password	varchar(128)	No	