Linkero

A Brand Monitoring Platform

This report is submitted in partial fulfilment of the requirements

for the

B.Sc. (Honours) Information Systems and Information Technology (DT249)

to the

School of Computing
College of Sciences & Health
Dublin Institute of Technology

Author: Stefano Richiardi
Student ID: D12124118
Supervisor:
Date: January 2019

Abstract

In this project I aim to build a data aggregator service for investigations and monitoring of counterfeit products online. The service will be implemented with a web interface for clients' access, an http server and specialized databases depending on data and performance requirements.

The core mission of this project is to try to simplify daily tasks of conterfeit investigations, and leverage server technologies to allow investigators to see the bigger picture painted when data from one case connect to one or multiple other cases previously unconnected.

Equally important will be the ability to deliver a product that is ready for production environment, that is: running on a virtual server accessible from the internet, protected from unauthorized intrusions and easily scalable should the user population grow significantly.

In this project I will also discuss pros and cons of data collection from online sources: from open API offered by onlines services to the community of developers, to unstructured data that can be scraped and schematized, and which legal implications need to be taken into account in light of the new European General Data Protection Regulation (aka GDPR).

Contents

Chapter 1

Introduction

1.1 Project background

The European Union Intellectual Property Office, in coolaboration with the OECD, estimated that in 2013 the value of imported counterfeit good was €338 billion, which corresponded to 2.5% of the total imports in the world trade [OECD(2016)]. The report also calculates that 5% of imported good in EU are counterfeit worth up to €85 billion. Counterfeit is an unfair commercial practice which takes advantage of the brand identity and presence on the market built by the brand owner, without incurring in the same costs of brand development (product design, quality standards and marketing). Other than a financial damage for the original brands, counterfeit products may pose safety risks for end consumers as much as for the people that work to produce them, since they evade the strict quality and safety standards set by national and international agencies around the world. From counterfeit iphone batteries that explode to counterfeit air-bags that do not trigger, they all can cause physical damage to consumers.

E-commerce has been rapidly expanding since it's early days in mid '90 when online marketplaces like eBay and Amazon were launched. Although it is hard to make a global estimate of market share for e-commerce companies, if we take the US market, the Census Bureau estimated that in 2009 online sales accounted for 4% of all retail sales, whereas in 2017 the e-commerce market share went up to 9% [of Commerce(2018)]. As legitimate retailers increase their presence online, shops selling counterfeit follow suit.

It is not surprising that many companies started to invest in fighting this trend, and that a good portion of the battleground is online. This tool is designed to help users to extract data related to a specific brands from online platforms such as eBay, Mercadolibre, Allegro, and present it in a tabular format.

Online marketplaces offer API open to developers in order to enable automated interaction with their platforms. This tool will focus on extracting sales and business registration data. The data will be stored in a database so that users can keep historical records of all their queries. At the same time, users will be able to leverage the growing dataset to link new investigations to old cases across all platforms whose data has already been stored in the database. For instance, serching the email address of a shop that deals counterfeit Diesel jeans may return details of multiple businesses registered on eBay and Mercadolibre at different times, as well as information about the administrator of a facebook page about counterfeit Raiban glasses. Users can use this tool to estimate brand damage caused based on the volume of sales, as well as a forensics platform to facilitate identity attribution of potential counterfeiters.

1.2 Background

Identifying online counterfeit items starts with reasearches of a brand or product online presence. The aim is to identify sellers that offer a branded product sold at a price point below average retail price. An investigator would ideally prioritize sellers with higher business turnover, and ideally located in jurisdictions where legal action is a dependable and impartial option. For instance countries like Russia and China, to name few, do not always offer adequate protection for European and North American companies, making it more difficul to pursue compensation from actors located withing their borders.

There are already different online companies that offer brand monitoring services: they range from keyword web-crawlers, to consumer sentiment analytics based on social platforms, to anti-counterfeit detection. We will review those companies and their services in details in the next chapter.

GDPR is the latest Europen Union legislation about processing of personal identifiable, privacy protected data belonging to european citizens. GDPR affects the implementation of this project in two ways: first, and most obvious, because users will submit limited personal data in order to be able to access Linkero services. Something as simple as an email address required to get notification for password recovery or data collection completed confirmation, is considered a PII (personal identifiable information) and thus is protected under GDPR. But GDPR affects also the collection of public personal details available online. This is the compliance requirement that is specific to companies providing *web scraping* services. When we deal with open source intelligence and attribution, PII are the most valuable data for anti-counterfeit investigations, therefore we will talk about how GDPR affects services like Linkero and what are the requirements to guarantee compliance.

1.3 Objectives

The objectives set for this project fall in two main categories: business and personal.

In relation to the first type, the goal is to build a business-ready web platform for brand-monitoring investigations. Being *business ready* means presenting a final application that runs on a virtual server directly connected to the public internet; that implements all industry starndard security features to guarantee that only authorized users can access its features and data; and is scalable, designed to accomodate an exponentially gowing number of users. The other aspect of the first goal refers to the ability to provide basic case management, data extraction and keyword searches capabilities tailerd for brand monitoring and anti-counterfeit investigations.

On a personal level, I aim at gaining a hands-on understanding of specific web technologies (e.g. JQuery, Django framework, NGINX, uWsgi), NoSQL databases (MongoDB, Redis), ansychronous and non-blocking programming techniques (multi-threading, AJAX, Pjthon Celery), development methodologies and testing practices.

1.4 Scope

This project will consist of a ready to deploy and use system for case management and web scraping data using eBay public API.

Regular users will be able to login, change password, set their preferred email address for general notifications and report delivery, launch queries, download data from completed reports, delete old queries, and perform keyword searches within all data collected and stored by any user in the database. Also the system will be configured to provide security protection against external unauthorized access and use of the system. Site administrators will manage the registration, password reset and deletion of users profiles.

1.5 Challenges and learning requirements

There are technical and personal challenges that have some kind of impact on the delivery of this project. To start with I am new to most of the technology used: from the basic configuration and administration of a Unix server, NoSql databases, client side web technology, SSL and DKIM protocols, etc.... This means that part of the time dedicated to the project will be spent bringing my knowledge and familiarity to a level that suits the project's requirements. Another element of technical challenge is determined by the use of a virtual server with only 1GB of memory and 1 single CPU. These limits are mainly dictated by financial reasons: this is the cheapest available VPS, which comes at $5.00 USD a month. Considering that I needed a VPS for the entire development and testing of the project, which took almost 2 years, anything more expensive would have impancted my family finances. The other problem is that limited memory and processing power started to show their impact and the code had to optimized accordingly.

On a presonal level I will have to take into account all the extra pressure that comes from being a father in a young family and professional responsibilities of a full time job.

1.6 Deliverables

The completed project will include:

- full documentation of the system design, development and functions;
- a working implementation of the system;
- a PowerPoint presentation of the project.

Chapter 2

Literature review

2.1 Introduction

This chapter will provide an overview of different topics that informed design decisions made during the initial stages of the project. We will start looking at the discipline of Open Source Intelligence, which refers to all those protocols and techniques used by government and private agencies to piece together intelligence reports using publicly available sources, since Linkero is a tool that facilitates the structured collection and analysis of a selected portion of open source data. The legal aspect of data collection and analysis will be explored with a reasoned summary of what is GDPR and how it impacts similar online services. Three sections will be dedicated to industry level best practices in relation to the security of servers facing the public internet, the usability of software interfaces and current guidelines to build reusable, maintainable and expandable code. Finally we will review briefly what are some of the current services already offering brand monitoring tools and how they differ from one another.

2.2 Open Source Intelligence

Open Source Intelligence (more commonly referred to as OSINT) is a relatively young discipline, that is concerned with the art of piecing together strategic intelligence from public sources of information. Michael Bazzel, a leading OSINT expert, defines it as:

> any intelligence produced from publicly available information that is collected, exploited, and disseminated in a timely manner to an appropriate audience for the purpose of addressing a specific intelligence requirement. [...] For the CIA, it may mean information obtained from foreign news

> broadcasts. For an attorney, it may mean data obtained from official government documents that are available to the public. For most people, it is publicly available content obtained from the internet [Bazzell(2015)].

As Bezzell explains, OSINT is not necessarily based on online sources, at least in its most broad definition. Journalistic style, old fashion dossiers filled with newspaper clippings are a form of OSINT. However it is fair to say that whenever OSINT is mentioned today, it will automatically produce the expectation that a large proportion of content is sourced through the internet. Another important author in this field, Stewart K. Bertram, explains:

> older OSINT research was limited by both the coverage of its information and the ability of the researcher to focus the capability on a specific subject, be it a person, location or topic. [...] What has changed this status quo is the arrival of the Internet, and particularly the explosion in the use of social media technology circa 2000. The rise of these two technologies created a multilingual, geographically distributed, completely unregulated publishing platform to which any user could also become an author and a publisher. [...] By increasing the coverage and focus of OSINT the Internet effectively promoted OSINT from a supporting role to finally sit alongside other more clandestine and less accessible investigative capabilities [Bertrom(2015)].

This explosion of sources of information causes another problem: reliability. Unless we are sourcing information from a scientific magazine which follows the rigorous fact checking protocol of most scientific researches, then we are facing a vast landscape of information with various degrees of veracity: reliable fact-checked sources on one side and *fake news* at the other end of the spectrum. The work of the OSINT investigator is to move in this virtually infinite universe of news, pick only relevant information and establish how reliable they are. Again this is not new, it is called *intelligence analysis*: "[...] the application of individual and collective cognitive methods to weigh data and test hypothesis within a secret socio-cultural context" [Hayes(2007)]. Information are scored from A (reliable) down to E (unreliable), plus F (reliability unknown).

Within the domain of counterfeit investigations, OSINT can have a number of roles to play. First off and foremost, should be used to establish if the items being sold are genuine or counterfeits. Secondly, it can be used to establish the extent of profits made (and therefore loss of income for the original brand) by the merchant selling them. And finally, OSINT can provide vital help in identifying the identity of the merchant as well as that of the manufacturer.

2.3 GDPR

The General Data Protection Regulation (EU) 2016/679 (a.k.a. GDPR) is the latest legislative effort of the European Union, that regulates the collection and use of European citizens' data. The two main goals that guided this new legislation were already set in 2015 by the Council of the European Union, which stated:

> The twofold aim of the Regulation is to enhance data protection rights of individuals and to improve business opportunities by facilitating the free flow of personal data in the digital single market [of the Council(2015)].

Essentially the GDPR gives more control to European citizens over the use of their data, thus restricting the ability of private and public organizations to process Personally Identifiable Information (a.k.a. PII) data, and at the same time it becomes a legislative framework that standardizes personal data processing in every country member of the European Union. Let's see in more detail what this entails and what are the consequences for a product like Linkero.

First off it is important to clarify some of the terminology used in the legislation. *Personally identifiable information*, or PII, is defined in privacy laws as one single piece of data that alone can identify a single person. For instance: the first and last name of a person, an email address, a physical address, a phone number, passport serial number, credit card number, a picture portraying an individual, a combination of username and password. GDPR applies to the collection, processing and storage of PII. A *Data Controller* is any organization that collects and manages PII belonging to European citizens. That includes online private companies (e.g. Google, Facebook the most obvious), as well as public institutions like primary and secondary schools and hospitals to name few. Another party that is involved in the handling of private data is the *Data Processor*, which in some cases coincides with the the Data Controller itself, when data is analysed internally, or with any other organization (e.g. contractors, vendors, suppliers, etc.) that receives PII from the Data Controller and processes it in its behalf. Finally the *Data Protection Officer* is a public office that is appointed to overlook and audit every Data Controller in its area of control.

What GDPR means for European citizens is that as of May 25th 2018, when it officially came into force, it grants much more rights over their own personal data, compared to previous legislation. These rights include *data portability*, which is to say that any Data Controller has to structure the collection and storage of PII in such a way that those information can be removed, transferred and re-allocated to another Data Controller without any further modification. In other words, this is a way to guarantee interoperability between Data Controllers, preventing single Controllers to lock-in their users. GDPR also states that EU citizens have the *right to information and transparency*, so that at any moment they can request their PII to be disclosed as they have been collected up to that point in time. Another right that generated a lot of discussion, is the *right to be forgotten*, which states that users can have their information permanently erased from a Data Controller, if they no longer require their services.

GDPR also sets more stringent constraints to the operation of organizations that fall within its scope. GDPR is self-declared *transnational in scope*, meaning that it applies to every company, anywhere in the world, as long as they hold EU citizens' PII data. Obviously, enforcement is an issue in cases such as companies that are based somewhere in the Cayman Islands. However this aspect is mainly meant at large multinational corporations, that could simply transfer their databases to another jurisdiction outside the EU in an attempt to circumvent its reach. Under the current definition of GDPR, those multinationals would still be subjected to EU law and would still be accountable as long as they have legal presence in any EU member state, regardless of where the PII data actually sits. GDPR also establishes a stronger *Data Subject consent*. This means that Data Controllers can no longer assume that their users' data can be processed and used for whatever purpose, just because PII were given to them during the registration process. Nor they can pre-tick the data consent acknowledgment box, or ask the user to tick a box only if they do not consent to the treatment of their data. On the contrary, Data Controllers have to explicitly ask and receive users' consent, and consent has to be renewed over time. Finally, GDPR requires organization to *report data breaches* within 72 hours after the breach was discovered. Data Controllers not only have to notify the supervisory authority and their users, but the notification has to carry adequate details to explain how the incident occurred, how many users are affected, and how is the organization going to respond.

With all these new rights and obligations in place, the last part of the legislation concerns to the *sanctions* for organizations that fail to comply. There are three levels of penalties that can be imposed:

1. a warning letter for the first time an organization is found in breach of GDPR, and if the breach is non-intentional;

2. a more serious breach could result in the order to commit to regular periodic data protection audits;

3. and finally, if the breach is deemed to be serious enough, GDPR prescribes two sets of financial fines depending of which obligations were missed by the Data Controller, the harsher of which could amount to up to €20 millions or up to 4% of the annual turnover estimated from the prior financial year, depending on

which is higher.

The question now is: how does GDPR affect online services that for their nature collect and process bits of PII that are publicly available? First and foremost, GDPR applies only to PII belonging to EU citizens, therefore we can say that an online service that collects PII can keep on providing its functions without any further thought as long as it stays clear of EU citizens. But let's say some users will be pointing the service to PII of European users, will that be sufficient to be found in breach of GDPR? The answer to this latter question is not a clear-cut *no*, there are actually use-cases where collection can still be an option. Obviously those cases do not include Data Subjects who expressed consent to their PII to be collected and stored by anyone online, that is not how data scraping for intelligence purpose works, especially in the context of open source investigations. Although it is too early for similar cases to have been tried, GDPR accepts that organizations may have a *legitimate interest* in scraping, storing and processing publicly available PII. In the specific context in which Linkero is designed, we are talking about an Intellectual Property owner, or an agent in their behalf, that has a vested interest in gathering information about an online shop that sells their products, under market value, or without being an officially approved re-seller. Having said that, GDPR definitely limits and restricts, for better or worse, the ability to extract, store and process public PII.

To conclude this section, it is interesting to mention how ensuring adequate privacy protection is a difficult balance and has its own trade-offs. This is the case of the WHOIS protocol maintained by ICANN (i.e. Internet Corporation for Assigned Names and Numbers), where internet domain registrants' PII are kept and made public. With the introduction of GDPR, personal details of the person who registered a domain will have to be hidden. Details of the domain registrant are valuable information in order to identify the person, or the signature of the person or group behind specific internet domains. This is particularly useful when security researchers try to attribute the ownership of phishing or spamming domains to a particular actor or group. Likewise, in counterfeit investigations, being able to tell how many and which domains were registered by the same actor is particularly useful, especially if you are trying to monitor the actor's activities. Information do not need to be accurate, in fact domain registrars do not verify the real identity of the person registering a domain. There are however details that need to be valid, like contact details (e.g. email address, phone number, ...) because on them depends the correct functioning of the domain, which is in the registrant's interest. The disappearance of those PII from WHOIS caused a great deal of frustration in the security community, and ended up with ICANN filing an official litigation against the European Union asking for some sort of exemption in order to keep offering relevant WHOIS data.

GDPR was created and voted by EU member states in a period when information is being considered the new oil of the global economy: access to personal data and consumer behaviour generates huge amounts of revenues for corporations like Facebook and Google. And the reason for that is simple: targeted ads campaigns are extremely effective and efficient, and marketing departments are willing to pay for that. The commercial application of petabyte of consumer data is well known. But there is another aspect to an unregulated access to data belonging to billions of people, which is *political influence*. As we have seen during the 2016 US election, bad actors managed to use personal data of millions of US citizens to both target specifically voters in swing states, and to analyze general voters' opinion in order to design very effective slogans. The same actors fabricated made up stories—literally, fake news—that would appeal to the most visceral human fears in order to influence undecided voters. That happened also during the British referendum of Brexit, and was attempted during the French presidential elections in 2017, and very likely in many other countries that did not receive as much media attention. At the same time we saw government bodies abusing their ability to tap into and collect indiscriminately data, just as Edward Snowden exposed how the US National Security Agency. GDPR may not be the best solution yet, but it is a much needed regulation at a time when personal information can have serious impact on the society.

2.4 Server security standards

A chain is as strong as its weakest link, and a system is as secure as its most vulnerable element, as the saying goes. In today's reality, any system facing the open and public internet has to be secured against people trying to misuse it. It is not just big banks and multinational corporations that have to put in place sci-fi grade defenses, but the same private wifi routers and connected thermostats have to be protected against bad actors that would gladly take them over and deploy them to perform fraudulent or criminal activities. It is well known that professional computer criminals constantly scan the internet to detect new connected devices and try to add them to their personal collection of bots for different purposes. Internet security is therefore a must regardless of the complexity of the system. Information Security literature identifies three main goals of a good system security strategy, also referred to as the *CIA triad* [Pipkin(2000)]: Confidentiality, Integrity and Availability. That means that a system security strategy aims at guaranteeing that data stored in the system will be accessed only by legitimate and relevant parties (confidentiality), that the same data will be kept in its original state away from sources that may alter, intercept or erase it (integrity), and that despite of the safety measures in place, if an attack succeeds, the data will still be accessible, and business will continue with minimal to no disruption (availability). In addition to the CIA triad, some experts also include *accountability*, which essentially means that if an incident occurs, the remediation team should be able to identify exactly where the breach happened and the location if not the identity of the person who executed the attack.

The industry identifies two main types of threat: internal and external. The external threat is probably the first and most popular to come to mind, often depicted like the hooded boy wearing a Guy Fawkes mask and typing away on his laptop: the hacker. Put simply, it is a person or organization that attacks a system from the outside. The other threat that does not get as much attention, but can be equally damaging, is the internal threat. The internal threat is represented by an internal user that intentionally or by mistake causes damage to the system. An attacker can be further classified based on his/her level of activity: a *passive* attacker tend to sit and intercept data exchanges, without affecting the system resources, whereas an *active* attacker directly alters system resources while the operation is underway [William Stallings(2015)].

Any attack normally impacts one or more of system assets: hardware, software, data or networks. There are five types of security measures that an organization can put in place to protect its assets: deterrent, prevention, detection, response and recovery. Deterrent measures are meant to scare attackers off before they even consider to start their exploits. The criminal law with its system of penalties is definitely the first deterrent that comes to mind. But media articles detailing how sophisticate and complex is a detection system can offer the same purpose. Preventive measures on the other end are technologies that anticipate potential attacks and stop them before any damage occurs. Firewalls are a great example. Since absolute prevention is rarely feasible, detection technologies can cover those situation in which an attack is found while still ongoing. Detection immediately triggers response measures, such as blacklisting an IP associated with a number of failed login attempts. Finally, recovery measures are the ones that are deployed after an attack occurred, and try to assess the damage and remediate where possible [Lampson(2004)].

Information security is a very vast and complex discipline, which directly reflects the complexity of modern interconnected systems. For the purpose of this project however we will only consider the basic security configuration required to protect a standalone Linux server.

In the next part of this section we will review in greater details some of the prevention technologies and solutions available, particularly those that were considered for this project, focusing on prevention, detection and recovery measures.

2.4.1 Prevention

Regular software updates

Applications running on the system are rarely bug-free. It is always good practice to keep every system application up-to-date with latest patches, since every time a new vulnerability is discovered, application developers tend to issue a new version or patch to close it.

Environment isolation

Rather than a technology, *isolation* is a principle and a policy that recommends every resource should be isolated from the rest, and access between them should be regulated and monitored. This is particularly critical in the context of systems that have public interfaces. Isolation applies to file management as well, requiring users files and folders to be isolated from one another, preventing one user's compromised credentials to be used to access everyone else's files. Finally, isolation applies to the same security mechanisms, since they should be isolated from the rest of the system, so that they cannot be compromised by the attacker [William Stallings(2015)].

Virtual Private Networks

VPN are a good solution to connect parts of the system that are not located on the same machine, or in the same network. Since communication between resources has to go through the public network, VPN offer a way to encrypt and protect data exchange based on asymmetric encryption technology.

SSL and TLS encryption

Similarly to VPN, SSL and TLS are two technologies that create an encrypted communication channel between a client and a server. The most common use case for these solutions is when login forms are required, so that a client can transfer login credentials to the server using a HTTPS connection, thus avoiding to transfer username and password in clear over a non-secure public network.

SSH

The Secure Shell, SSH, is the default application for remote server management. Like VPN and SSL, it establishes an encrypted connection between a client and a server, relying on asymmetric encryption. It allows system administrators to login remotely to the server command line shell. It can be configured to either accept a standard password login, or to accept automatic logins from approved clients only.

Firewalls

Firewalls can be described as *intrusion prevention systems*. A firewall is an application that filters all incoming and outgoing network traffic between the internal network and the public internet, and responds to a set of rules to decide was is allowed and what is not. It provides a single choke point for network traffic, where the system can act as detection mechanism for suspicious connections, as well as traffic logger for audits. There are four different types and corresponding level of complexity: packet filtering, stateful inspection, application-level gateway and circuit-level gateway. The last two are the most sophisticated and require also a lot of CPU and memory resources, for this reasons are not suitable for this project [Cheswick(2003)].

Packet filtering is the most basic capability of a firewall, and bases its decision on whether to allow or reject a connection on the information contained in an IP packet: source and destination IP, source and destination transport-level address, IP protocol and interface. Packet filter firewalls however are incapable of establishing when a connection to a port number above 1024 is legitimate or not. This is where a stateful inspection firewall comes into play. Stateful packet inspection works by keeping track of ongoing connections, session numbers, and originating party in order to prevent attacks like session hijacking [William Stallings(2015)].

Users and groups least privileges

Just like environment isolation, assigning users and groups with the right amount of privileges and permission is more a policy than a technology. This policy states that users should be given the minimum amount necessary of privileges in order to perform their tasks. This way if an account is compromised, there is a chance that damages can be controlled and limited. Permissions include the ability to read, write, execute, delete or share a specific resource. More advanced resource and permission management policies include a temporary aspect, in that permissions are granted for a limited amount of time, and are active only during working hours, or whenever a specific task needs to be performed [William Stallings(2015)].

Password policy

A password policy is design to make sure that all users comply with the industry best known standards of password security. A password policy should establish:

- the strength of the password by insuring and checking that fa user generate password uses the most varied combination of lower and upper case letters, numbers and special symbols in a string of a minimum length in order to guarantee that a brute-force attack (attempting to guess the password trying every possible combination) would require a significant amount of time to succeed;
- the lifetime of a password, which should be long enough not to cause user friction, but short enough to guarantee that should the password be compromised, there is a chance it will be too late to use it before expiration;
- a new password cannot be the same as any previously used;
- a set of alternative contact details and security questions that should be used to recover a lost or compromised password;
- a maximum numbers of failed login attempts, after which the account is locked and a password reset is required;
- recommendations about the right way of storing the password: making sure the password is not written down or saved in an unsecured file.

Encryption of data at rest

Systems that store data in databases may be compromised, when that happens if the data is stored in clear, the attacker can make a copy of the database, and access all the contents. Data that is not loaded in memory can be encrypted, so that simply making a copy and moving it outside of the current system, will not allow the attacker to access it, lacking the necessary decryption keys.

Physical security

Physical and infrastructure security is sometimes an overlooked aspect, when the focus is all on hardening the software and data, when it is actually as important. There are three main categories of threats to physical infrastructure: environmental, technical and human caused. Environmental threats includes both exceptional events like tornadoes, floods or earthquakes, as well as less traumatic events like the level of humidity, dust, external temperature, gas and other chemical elements, factors that in the long run can actually cause serious damage. Technical threats include power surges or drops, and electromagnetic interference. Finally the most common human-caused threats are: unauthorized physical access, vandalism, theft and misuse.

For the purpose of this project however, no countermeasures were taken to mitigate those threats, since the system is composed of a single virtual server, run by the Virtual Private Server provider Linode ltd. We assume that safety and security measures are being adopted by the service provider [William Stallings(2015)].

2.4.2 Detection

Intrusion detection systems

Intrusion detection system are the second level of defence, in case all of the previous preventive measure fail to stop an external intruder. The basic idea behind an intrusion detection system (IDS) is that the way an intruder interacts with the system is substantially different from any other authorized user, thus his/her behaviours can be described in quantitative terms (the intruder's signature) and this description can then be implemented in a system whose purpose is to ring an alarm when such behaviour is detected and block the intruder before any damage occurs [William Stallings(2015)].

These are the main types of IDS currently available:

Host-based IDS : application that runs on a specific host, and only monitors activity on that host.

Network-based IDS : application that runs on a host part of a larger network, that monitors all the activities within that network.

Distributed or hybrid IDS : application that gathers data from a cluster of sensors distributed across a network [Shirey(2000)].

There are essentially two ways that an IDS can be programmed to detect intruders: we can either decide that a range of behaviours is legitimate, therefore everything else that deviate from that norm is an intruder, or we can list bad behaviours, and tell the IDS to trigger the alarm when any of the behaviours listed is detected. The first strategy is called *anomaly detection* the latter *signature detection*. Without going in too many details, anomaly detection is clearly the most effective of the two, and would be anyone's first choice if not for the fact that is very difficult to define a norm. So difficult that it requires very sophisticate approaches like statistical analysis and machine learning techniques [Garcia-Teodoro(2009)]. Signature detection on the other hand is a bit easier to implement, and guarantees a lower number of false positives, given that its database is stored with good quality intruders' signatures. The problem is that any new, unexpected and undefined intrusion behaviour will not be detected, leaving the system vulnerable to zero-day attacks [William Stallings(2015)].

Regular file, service and vulnerability audits

A security auditing system is a very complex set of hardware and software tools, policy, processes and old fashion human labour. Such a system is beyond the scope of this project, but it is worth briefly mentioning it, for completeness sake, within the discussion on information security.

A well designed security auditing system has the following requirements:

- being able to establish at any time whether a system's security is intact or has been breached;
- capture by default all data that is necessary to establish if an intrusion occurred, when, how and possibly by who;
- store relevant data for future analysis and forensics;
- feed anomaly signatures back to an intrusion detection system signature database [William Stallings(2015)].

An audit can be triggered right after a potential attack is detected as well as can be repeated periodically. An audit can be triggered in different ways:

basic alerting : is the simplest for of trigger, which only requires a single rule to trigger an alarm if a specific event occurs.

Baselining : is the practice of defining a normal versus unusual event or pattern. This feature overlaps with an IDS signature-based trigger.

Correlation : is taken from statistical analysis and consists on triggering an alarm or notification when an event or set of events is used to infer the presence of a potential breach [William Stallings(2015)].

2.4.3 Recovery

Business continuity plan

A business continuity plan is a set of measures, mechanisms that ensure an organization can go on or quickly restore basic and critical operations with minimal downtime [Wood(2010)]. In a business continuity plan a series of scenarios are identified, for instance: loss of workspace, loss of personnel, loss of network infrastructure. Every scenario is coupled with a list of function critical resources that can be impacted and relative countermeasures to be executed along with roles and responsibilities assigned to specific people. Having a backup of all the data on the system is a preventive measure, using it to restore data that was lost, encrypted or corrupted during an attack is what recovery looks like.

2.5 Design Engineering

Goal of a good software project is to produce software that not only works, but that is easy to understand, to test, maintain and expand if needed. Helwett-Packard developed five attributes [Grady(1987)] that define the quality of a software product:

Functionality concerns the set of functions and capabilities of the program, as well as its security.

Usability relates to how well the intended users are able to operate the program.

Reliability is the measure of frequency and severity of the program failures.

Performance indicates the program speed of response, resources consumption, efficiency and throughput.

Maintainability is concerned with how the software is easy to troubleshoot, adapt, configure and extend.

Design engineering takes care of the quality of a software, and is the last stage of software modelling before code development can actually start [Pressman(2005)]. Design engineering is divided in four elements: data design, component-level design, interface design and architectural design. We will briefly explore each one here.

2.5.1 Component-level design

Component design refers to the procedural aspect of a software program: functions, processes and algorithms are the object of this element of design. Design engineering developed over the years a set of rules and guidelines that allow developers to write code that is maintainable and extendable:

Modularity states that the final software should be the result of separate nuclear components, called modules, that will be integrated and interact with each other in order to produce the desired functionalities [Meyers(1978)].

Information hiding suggests that, as much as possible, the data used within one module for its operations, should be hidden from other modules. Only the bare minimum amount of information should be exchanged between modules to ensure software operations [Parnas(1972)].

Functional independence is a concept directly derived by the previous two, and it says that a module should focus on a specific functionality (measured in level of cohesion) and it should have as little interaction with other modules as possible (expressed in terms of coupling) [Parnas(1972)].

Refinement rather than a prescription this is a top-down process of defining each element of the software starting from a high level of abstraction, and progressively narrowing down to more detailed procedural levels [Wirth(1971)].

Refactoring is another technique that aims at simplifying the design of a program without changing its functionality. The code is examined for unused elements, redundancy, inefficient or unnecessary algorithms [Pressman(2005)].

Robert Martin took those rules and techniques and elaborated further into his five S.O.L.I.D. principles [Martin(2003)]:

Single responsibility states that each module or class should be responsible for one function, and that function should be well encapsulated within the module or class. This principle combines the concept of cohesion and information hiding.

Open-Closed dictates that a module or a class should be built in such a way to prevent modification, but still offer an interface that allows it to be inherited and expanded. The reason is that, assuming a new requirement arise calling for a new functionality to be implemented, adding that functionality to an established class could alter its behaviour and cause unintended consequences in other parts of the code where that class is in use. Open-closed principle suggests instead a new subclass should be created in order to fulfil the new requirement, thus guaranteeing the stability of an application, while allowing for functionality extension.

Liskov substitution is an extension of the *open-closed* principle. It requires that any child or grand-child class can take its parent's class place in the application without altering the general functionality of the application. This principle is intended to guarantee consistency of interface within class types of the same family.

Interface segregation recommends to develop simple client-specific interfaces, rather than one general purpose interface for every known client. One general purpose interface would include all functionalities that are relevant to any client, thus forcing every client to implement functionalities that are not necessarily relevant. The other problem is that if a new requirement calls for a new interface to be added, all pre-existing clients should update their implementation to fit the new interface, whether they support it or not. To solve this issue, single functionalities should have their own interface, and clients can implement selectively just those that are relevant.

Dependency inversion finally integrates the concept of module de-coupling, stating that higher level modules should not depend on lower level ones. The problem with modules dependencies is that a change in one could force changes to its dependees, the assembly of modules could result is expensive operations to guarantee that every dependency is respected. This goes against module reusability. The solution is to add a standard interface between two modules. A good example to explain this principle is imagining the relation between a lamp and the home electrical supply: making them directly dependent equates to soldering the lamp wires to the electrical system, instead the electrical socket was developed and became the standard interface between any electrical tool and the electrical power supply [Pressman(2005)].

2.5.2 User interface design

User interface design is concerned with making sure the user will be able to actually use the software, by designing an interface that is meaningful and intuitive. Theo Mandel developed three *golden rules* [Mandel(1997)]:

Place the user in control means that the interface has to be designed with the user's needs in mind, not the developer. And these are some of the commands user's needs: a user needs to be able to perform the same action in different ways (e.g. via console commands, mouse action, keyboard short cuts, etc.); a user sometime needs to suspend a task and move to a new one, before resuming the old one, as well as being able to *undo* an action made by mistake; a user needs adjust its interaction according to his/her level of expertise (e.g. initial point and click can be replaced by macro recording of a sequence of actions); a user does not need to see nor interact with low level functions; a user needs to manipulate objects of a visual interface as he/she would do if those objects were a physical thing.

Reduce the user's memory load requires for interfaces to be intuitive, without much study a user should be able to know what each part does just by looking at it. Mandel [Mandel(1997)] further specify this rule, by saying that the visual layout should be modelled based on a real-wold metaphor using well established visual cues; the initial default configuration should make sense for the average user, allowing the expert user to change it to his/her liking; information should be presented in a progressive way, starting from general task, down to more detailed functions.

Make interface consistent demands that a standard layout should be defined and kept consistent throughout the rest of the program. Context layers are very important, allowing the user to know at a glance where his/her current task lays within the general process (e.g. window tiles, graphical icons, color coding, etc.), how the user was able to access the current interface and what are the available next steps. Interface consistency also requires that design takes interactive sequences that have become standard (e.g. *Cntrl-C* is the standard for copying pieces of text) and makes them available in the application.

2.5.3 Software architecture

Software architecture encompasses two elements of software design: data design and architectural design. Here we will see the main concepts that are used to describe data and architectural design.

Data design

Just like this project, most recent software applications and services have been developed around the idea that accessing, storing and computing large amounts of data can offer innovative solutions. Which is a way of saying that data today is the new black gold, it is not a surprise than that designing what kind of data will be analysed and how it will be stored and structured is extremely crucial [Pressman(2005)].

At architectural level, data design is mainly concerned with the structures that will contain the data and how those integrate within the overall architecture. We are talking about using external databases to store the business data. Depending on the size and complexity of data requirements, we may see a small amount of centralized databases that store production data that is needed to provide live core business services. With bigger corporations and a growing number of diversity of database structures, it may become more difficult for data analysts to access information that is spread across multiple databases with multiple diverse data structures. *Data warehouses* come to the rescue. Data warehouses essentially store copy of production business data in a separate environment, and make sure that all the data is kept using the same structure, thus allowing data analysts to perform data mining on current and historical data, in order to offer strategic insight for business executive decisions [Mattison(1996)].

Architectural design

Architectural design aims at building a blueprint of the entire system, with special emphasis of how each component interacts with the others. There are two aspects that describe an architecture: style and pattern [Pressman(2005)].

Style refers to the shape of the overall system structure, it is a static representation of the system: which components interface directly to each other, which are peripheral, which are central, and so on. Examples of architectural styles include:

Data-centered architecture refers to a system that is designed around its database, where all other components are clients that exchange information only with the central server, with little or no interaction with each other.

Data-flow architecture instead is applied to a system where data moves mainly in one direction, first as input, leaving the system as output, after all the modules located in-between these two points have modified somehow the original input.

Call and return is a style in which modularity is central to the overall operation, where each function is broken down in a hierarchy of modules. This style has the advantage of being relatively easy to modify and scale.

Layered architecture, finally, describes a system in which components are grouped from an outer layer, normally the user interface, through an application and utility layer, down to the core layer which performs machine levels operations.

These are just a sample of the most common architectural styles [Pressman(2005)].

An architectural pattern instead describes the most dynamic aspect of the system: how each component interacts with the others, how they behave. Architectural patterns are also referred to as a solution to a standard software development problem. These are some examples of patterns and the problem they solve:

Concurrency is the ability to perform multiple, different and independent tasks at the same time. One problem developer face is when the system has limited resources, like one single CPU, so that real multitasking cannot be achieved. An *operating system process management* pattern provides that concurrency can be simulated by assigning a turn to each task for a limited amount of runtime. This solution offers a real advantage for tasks that depend on many read and write operations with external slow components.

Persistence is the ability to store data, after the process that created it is finished. *Application level persistence* pattern for instance provides that applications show generate and manage their own file format where to store user-generated data or software configurations. Another pattern that offers a solution is the *database management system*.

Distribution addresses the issue of components communication in a distributed environment, like clients exchanging emails: each client cannot assume that the recipient will be online in order to receive the communication. The *broker* pattern's solution consists in building an agent to delegate to a communication, and the agent will guarantee the message delivery by waiting for the recipient to be ready to receive the message [Bosch(2000)].

2.6 Existing brand monitoring services

Loren Ipsum ...

Chapter 3

System implementation

3.1 System architecture

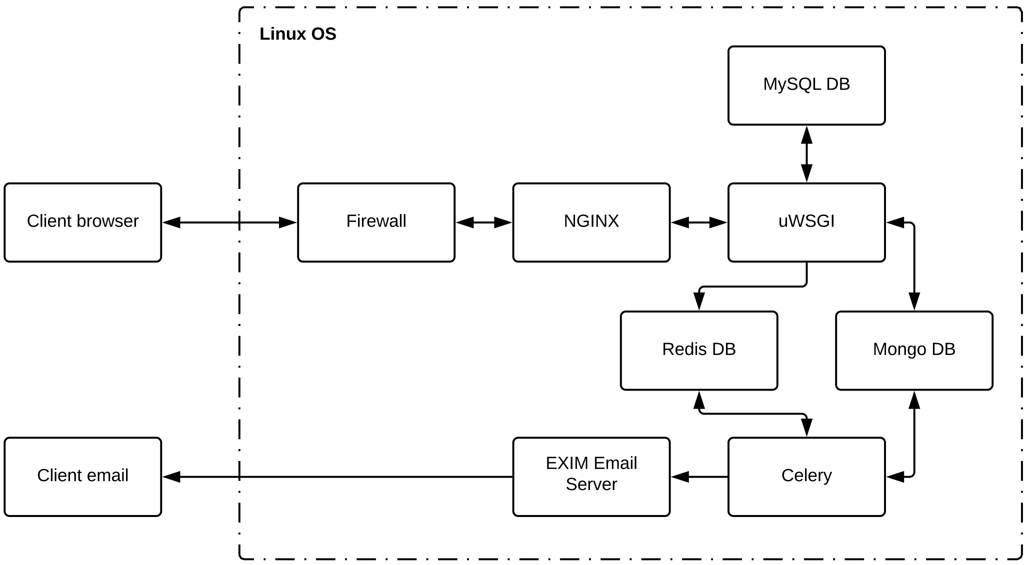

Figure 3.1: Linkero system architecture

3.2 Technologies

3.2.1 Server operating system

3.2.2 Web application framework

3.2.3 Databases

3.2.4 Web engine

NGINX was chosen to serve the pages of this application.

3.2.5 Firewall

3.3 Development methodology

3.4 Requirements

3.4.1 Functional requirements

3.4.2 Non-functional requirements

3.4.3 User requirements

3.4.4 Requirements analysis

3.5 Use cases

3.6 Sequence diagrams

3.7 Code samples

Chapter 4

Testing and deployment

4.1 Unit testing

4.2 Integration testing

4.3 System testing

4.4 Usability testing

4.5 Deployment

Chapter 5

Evaluation

5.1 Lessons learned

5.2 Future work

5.3 Conclusions

5.4 Deployment

Bibliography

[Bazzell(2015)] Michael Bazzell. *Open Source Intelligence Techniques*. Amazon.co.uk Ltd, 2015.

[Bertrom(2015)] Stewart K. Bertrom. *The Tao of Open Source Intelligence*. IT Governance Publishing, 2015.

[Bosch(2000)] J. Bosch. *Design & Use of Software Architectures*. Addison-Wesley, 2000.

[Cheswick(2003)] W. Cheswick. *Firewalls and Internet Security.* Addison-Wesley, 2003.

[Garcia-Teodoro(2009)] P. Garcia-Teodoro. *Anomaly-Based Netowrk Intrusion Detection.* Computer & Security, Vol. 28, 2009.

[Grady(1987)] R. B. Grady. *Software metrics: establishing a company wide program*. Prentice Hall, 1987.

[Hayes(2007)] Joseph Hayes. *Analytic Culture in the U.S. Intelligence Community*. Ed. Center for the Study of Intelligence, 2007.

[Lampson(2004)] B. Lampson. *Computer Security in the Real World.* Computer, 2004.

[Mandel(1997)] Theo Mandel. *The elements of user interface design*. Wiley, 1997.

[Martin(2003)] Robert C. Martin. *Agile Software Development, Principles, Patterns, and Practices*. Prentice Hall, 2003.

[Mattison(1996)] R. Mattison. *Data Warehousing: Strategies, Technologies and Techniques*. McGraw Hill, 1996.

[Meyers(1978)] G. Meyers. *Composite Structured Design*. Van Nostrand, 1978.

[OECD(2016)] EUIPO OECD. *Trade in Counterfeit and Pirated Goods: Mapping the Economic Impact*. OECD Publishing, Paris, 2016.

[of Commerce(2018)] US Department of Commerce. Quarterly retaul e-commerce sales. *US Census Bureau News*, 2018.

[of the Council(2015)] Presidency of the Council. Proposal for a regulation of the european parliament and of the council on the protection of individuals with regard to the processing of personal data and on the free movement of such data (general data protection regulation). *Official Journal of the European Union*, 2015.

[Parnas(1972)] D. L Parnas. On criteria to use in decomposing systems into modules. *CACM*, 14(1):221–227, 1972.

[Pipkin(2000)] Donald Pipkin. *Information security: Protecting the global enterprise.* New York: Hewlett-Packard Company, 2000.

[Pressman(2005)] Roger S. Pressman. *Software engineering*. McGraw-Hill, 2005.

[Shirey(2000)] R. Shirey. *RFC 2828 Internet Security Glossary.* Internet Engineering Task Force, 2000.

[William Stallings(2015)] Lawrence Brown William Stallings. *Computer Security, principles and practice.* Pearson Education Ltd, 2015.

[Wirth(1971)] N. Wirth. Program development by stepwise refinement. *CACM*, 14(4): 221–227, 1971.

[Wood(2010)] T. Wood. *Disaster Recovery as a Cloud Service Economy Benefits.* Proceedings USELINUX HotCloud 10, 2010.