Abertay University

School of Design and Informatics

BSc (Hons) Ethical Hacking 2020/21

Evaluating the Effectiveness of Various Developments to the Contactless 'Active' Reconnaissance Process

---

Word Count: 35,654

Author:
Liam Shillinglaw

Supervisor:
Jamie O'Hare

ABSTRACT

Our global reliance on the internet has exponentially accelerated due to the Covid-19 pandemic, leading to a drastic increase in the sheer scale and diversity of exposed Internet-connected devices. Many of which are prone to frequent unprovoked attacks by malicious threat actors. The use of the contactless 'active' reconnaissance (CAR) process is an appropriate non-direct method for tackling the problem of identifying and assessing the ever-expanding global attack surface. However, current tooling implementing the process lacks in both functionality and usability. This paper explores various avenues to improve upon the effectiveness of this process. The overall aim was to design, produce and evaluate a non-intrusive asset discovery and vulnerability assessment tool capable of effectively identifying known-vulnerabilities in exposed Internet-connected devices.

To accomplish this project, a novel tool named 'Informant' was produced, expanding upon the functionality of a varying array of freely available passive data sources, resulting in powerful asset discovery and risk assessment capabilities. The tool is an ideal solution for identifying low hanging fruit, such as services susceptible to known-vulnerabilities and dangling DNS records. The developed solution possesses many attributes commonly associated with active scanning solutions but without the direct active scanning element. Features include: Common Platform Enumeration (CPE) construction, Common Vulnerability and Exposures (CVE) assignment, default configuration CVE identification, rDDoS predictions and risky port assessments. The developed solution reveals how the correlation and merging of multiple Internet-wide scanning and passive DNS (PDNS) sources can band together to create a tool capable of competing with enterprise asset discovery and vulnerability assessment solutions.

During initial experimentation, Informant recognised 953 services across 4 UK-based universities, identifying 5,515 potential known-vulnerabilities. 48% of these known-vulnerabilities were identified as potentially being vulnerable on their default software configurations. Several other experiments within the paper explore and compare further discovery and risk assessment capabilities of Informant against industry-standard vulnerability assessment tooling. A notable experiment found Informant to have a known-vulnerability detection effectiveness of 75% compared to OpenVAS, an improvement over Scout, a competing CAR tool. Informant is also found to be superior in terms of usability, capabilities, and accuracy when compared to freely available CAR tooling solutions. Further studies could look to improve the default configuration CVE identification and CPE construction approaches implemented by leveraging the power of machine learning.

Acknowledgements

Throughout the entire process of my dissertation, I have received a great deal of support from family, friends and industry professionals to whom I would like to express my utmost gratitude. Firstly, I would like to thank Jamie O'Hare for his ongoing supervision, guidance and willingness to give up his time throughout this project. Secondly, I would like to acknowledge Tiago Henriques and Brian Martin for their ongoing work and professional insight. Finally, I would like to thank Kara Herzog and the rest of my family and friends for their continued support and contributions towards this study.

Contents

List of Figures

List of Tables

Abbreviations, Symbols and Notations

API	Application Programming Interface
ASM	Attack Surface Management/Monitoring
BGP	Border Gateway Protocol
CAR	Contactless Active Reconnassiance
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposures
CVSS	Common Vulnerability Scoring System
DDOS	Distributed Denial of Service
DNS	Domain Name Service
FS	Formatted String
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
ICMP	Internet Control Message Protocol
ICS	Industrial Control Systems
IPO	Input-Process-Output
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IWS	Internet-wide Scanning
JSON	JavaScript Object Notation
ML	Machine Learning
NER	Named Entity Recognition

NIST	National Institute for Standards and Technology
NLP	Natural Language Processing
NVD	National Vulnerability Database
OSINT	Open-source Intelligence
PDNS	Passive Domain Name System
PLC	Programmable Logic Controllers
PoC	Proof-of-Concept
rDDoS	reflected Distributed Denial of Service
SCAP	Security Content Automation Protocol
ShoVAT	The Shodan Vulnerability Assessment Tool
SSH	Secure Shell
URI	Uniform Resource Identifier
VMS	Vulnerability Management System
WFN	Well Formed Name

CHAPTER 1

1 INTRODUCTION

1.1 BACKGROUND AND CONTEXT

Our reliance on Internet-connected devices has exponentially accelerated in part to the global Covid-19 pandemic, forcing a large percentage of the world's citizens to work remotely for a prolonged period (Feldmann *et al.*, 2020). The overwhelming adoption of home-based working has led to an increased attack surface for many organisations, triggering a substantial rise in malicious cyber-related incidents across all sectors. According to Morgan (2020), by 2021 such incidents will cause the world over six trillion dollars annually, making cybercrime the single most significant financial threat to both individuals and organisations. This substantial expense, coupled with the exponential increase in reliance on Internet-connected devices in light of the Covid-19 pandemic, has dramatically attributed to businesses and individuals calling for a more serious stance on cybersecurity (Wright, 2020). Consequently, this has resulted in a demand for the emergence of Attack Surface Management (ASM) solutions aimed at identifying and auditing the exposed attack surface of organisations (Lidome, 2020).

The most common types of attacks are relatively trivial to perform, taking advantage of known-vulnerabilities within individual components and services running on devices. These often underestimated, already identified, and disclosed flaws are known to be susceptible to vulnerabilities that can be mitigated with a simple software patch (Frei *et al*., 2006). However, it remains the case that the most successful, well-publicised attacks could easily be prevented if correct device hygiene practices were adhered to. Gartner believes that 99% of all exploited vulnerabilities are known for at least a year before exploitation in the wild (Panetta, 2016), emphasising the need for adequate device hygiene practices to prevent known vulnerabilities. In recent times, a critical remote desktop protocol (RDP) vulnerability disclosed and patched by Microsoft in the early second quarter of 2019 was still found to be widely exploitable, seventeen days after the initial disclosure and patch (Grangeia, 2019). As illustrated in Figure 1, a staggering total of almost one million devices were still found to be vulnerable during this time.

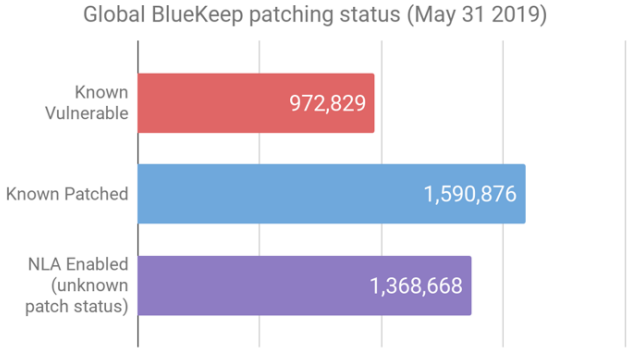


Figure 1: BlueKeep IWS Statistics (Grangeia, 2019)

From an abundance of academic research, it is well established that the entirety of the internet can be scanned in a mere matter of minutes. Several reputable Internet-wide scanning (IWS) platforms exist to accomplish this, all with differing capabilities, such as Internet Protocol version 6 (IPv6) scanning support, amongst others. The likes of the Censys (2021), Shodan (2021), and BinaryEdge (2021) platforms all conduct reconnaissance on the entirety of the Internet Protocol version 4 (IPv4) namespace, enumerating and gathering information on devices. These scanning platforms are excellent tools for discovering and monitoring the security of Internet-connected assets. However, they can also be a rich source of information for malicious users. As such, tech-savvy organisations deploy a multitude of countermeasures to avoid being identified by IWS probes - a technique commonly known as security via obscurity within the cybersecurity industry (Patton *et al.*, 2014). This, in turn, makes IWS platform results less reliable, as assets can evade IWS platform scans. Therefore, it is fair to assume that correlating data from multiple Internet-wide scanners, although tedious, would significantly mitigate this problem due to the complexity of identifying and blocking differing sophisticated benign scanning probes (Shori, 2018).

The capabilities of IWS are seemingly endless, with a continuous flow of research revolving around and expanding upon the concept. Notable research is emerging investigating how IWS platforms can be used to stealthily conduct reconnaissance, known as Contactless 'Active' Reconnaissance (CAR). Researchers have investigated how CAR data can be used to identify services running that are susceptible to known-vulnerabilities based on Common Platform Enumeration (CPE) data. A leading researcher in this field developed a Python-based CAR tool known as 'Scout' to evaluate the security perimeter of UK universities' (O'Hare, Macfarlane and Lo, 2019).

Despite the evident increase in Internet-connected devices and the abundance of academic research in this area, few practical solutions currently exist that expand upon the CAR process. Additionally, most tooling that is available is command-line-driven and non-intuitive for the average user. The only notable tooling exception is by Rodrigues (2019), who designed and developed a fully automated passive asset discovery tool with an intuitive graphical user interface. However, the program has never been distributed and remains closed source.

These current works allude to many future work avenues, including correlating multiple IWS data sources, machine learning (ML) based data enrichment and unique asset pivoting via external passive data sources. Combining these techniques to develop an intuitive user-friendly solution would greatly benefit institutions in passively managing and monitoring the security of their ever-growing inventory of Internet-connected devices.

1.2 Aims, Research Questions and Objectives

This project aims to design, produce and evaluate a non-intrusive asset discovery and vulnerability assessment tool to identify known-vulnerabilities in exposed Internet-connected devices effectively. This broad aim can be deconstructed into four concentrated research questions:

1. How can IWS platforms offer a reliable and comprehensive source of rich passive data?

2. How can this rich data source be used to generate accurate, unique identifiers to enable service banner identification?
3. What effectiveness does enrichment possibilities provide to accurately identify known-vulnerabilities?
4. How can further reconnaissance methods contribute to the identification of known-vulnerabilities in exposed Internet-connected devices?

The aims and questions posed will be achieved by completing the following objectives:

1. Carry out a comprehensive literature review, critically analyse prior research work by means of comparisons in the fields of Internet-wide vulnerability scanning along with passive known-vulnerability detection.

2. Based on knowledge derived from the literature review, design and develop a clear and concise methodology and tool capable of accurately categorising and identifying vulnerable services based on data sourced from IWS platforms.

3. Conduct experiments using the procedures highlighted in prior research. Analyse, improve, and revise the procedures if feasible.

4. Analysis, compare and evaluate the results of the experiments in line with the overall aim of the project, previous research papers and where needed, enhance existing methods.

1.3 SCOPE

Unlike previous research in this area that has obtained privileged access to large IWS datasets and extensive financial backing from fortune 500 entities, this dissertation will be solely conducted using free publicly accessible information. Tooling that solely utilises non-passive techniques directly from the host system will not be included in this paper, neither will platforms that merely perform on-demand scanning. All literature relating to the research questions are within the scope of this project.

1.4 STRUCTURE

Proceeding the introduction section to achieve the outlined objectives of this dissertation, the report is structured as follows:

• **Chapter 2 – Literature Review**

- This chapter establishes and critically analyses prior research work in the fields of Internet-wide vulnerability scanning along with passive known-vulnerability detection.

• **Chapter 3 – Methodology**

- This chapter will define the foundations for the methods and processes used to design, develop, and evaluate a tool capable of meeting the outlined objectives. By utilising and refining well-established techniques from critically analysed academic research. This chapter will also include and discuss the motive behind key design decisions and highlight steps taken to implement the outlined design.

• **Chapter 4 – Results**

- This chapter presents, highlights, and analyses the findings produced from experiments outlined in the methodology chapter.

• **Chapter 5 – Discussion**

- This chapter discusses and evaluates the success of the developed solution and the results produced against related academic work where feasible.

• **Chapter 6 – Conclusions**

- This chapter summaries the entirety of the dissertation, detailing key findings and highlighting the strengths and shortcomings of the project, in addition to outlining recommendations for further research.

CHAPTER 2

2 LITERATURE REVIEW

2.1 INTRODUCTION

This chapter aims to review relevant cybersecurity literature critically and is separated into four sections: Internet-wide Scanning, Current Vulnerability Assessment Tooling, Service Banner Analysis, and Passive Data Enrichment. With a vast array of sources available covering the topic of IWS and vulnerability assessment, this literature review is limited to credible sources, primarily consisting of technical and academic papers. As stated within the project scope, tooling that solely utilises non-passive techniques directly from the host system will not be included in this review, neither will platforms that merely perform on-demand scanning. The Current Vulnerability Assessment Tooling section will investigate and evaluate the present state of freely available tooling, expanding upon concepts presented in the above topics. Service Banner Analysis methods will be compared and analysed based on available literature. Finally, the Passive Data Enrichment section will examine academic literature covering alternative Open-source intelligence (OSINT) sources.

2.2 INTERNET-WIDE SCANNING

The concept of IWS has quickly proven to play a fundamental role within security research and the broader cybersecurity landscape. The primary objective behind IWS is simple; to identify and evaluate the presence of Internet-connected devices. Recently an abundance of research is emerging looking into innovative potential uses for the vast array of data internet scanners gather. Notably, avenues this technology has opened range from widespread service vulnerability detection to estimating reflected Distributed Denial of Service (rDDoS) throughput, as well as predicting possible targets of zero-day attacks (Halderman, 2013; Durumeric *et al*., 2015; Leverett and Kaplan, 2017). A recent paper by Leverett, Rhode and Wedgbury (2020) has paved the way for new ground-breaking research in the field of forecasting future vulnerabilities. An impressive achievement with an endless number of practical applications within the security industry, indicating that there are still many more additional avenues to explore within this research area.

Extensive academic work in this area led to numerous cloud-based IWS search engines, such as the popular Censys and Shodan platforms. These platforms, among others, are dedicated to scanning the entirety of the internet at regular intervals, searching for, identifying, and enumerating active services on responsive publicly facing Internet-connected devices. Theoretically, these platforms can detect every active device exposed to the internet, making a dangerous but valid security practice known as security through obscurity ineffective (Patton *et al*., 2014), stipulating that a more proactive approach should be taken towards device security. Further reinforcing that if devices can be detected, it is just a matter of time until they are attacked (HackerTarget, 2019). Shodan is arguably the most well-established IWS

platform in the security industry and has been widely publicised by CNN as being “the scariest search engine on the Internet” (Goldman, 2013), and with good reason. The platform truly pushed the bounds of technology at the time of its initial release and set the stage for future research work in this area. However, the CEO of Shodan, John Matherly, never initially intended for the platform to be used in the way it is today. In a recent interview with PortSwigger, he voiced that “the use case I designed Shodan for was market intelligence, not security” (Pritchard, 2020). Showcasing just how far the platform has evolved since its conception, where its original intention was to monetise the gathering of marketing statistics, enabling organisations to visualise how their software products were being implemented by consumers across the internet (Pritchard, 2020). In recent times, Shodan has gained some unwanted publicity as a result of two attackers utilising the platform to identify over 800,000 vulnerable printers. A relatively small portion - 50,000 - were later exploited, printing propaganda material in support of a notorious YouTube channel (Brewster, 2018).

In the grand scale of things, the Shodan printer scandal is insignificant. Leveraging the full potential that IWS provides opens many more avenues for vastly more sophisticated attacks and security methodologies to be formed, as previously drawn upon. Due to the evident potential use of data gathered from IWS scanning being malicious, a considerable amount of time and thought has gone into analysing the broader impact of scanners. Shori (2018) investigated how blocking Shodan scans affected the attack volume on a device. Although network volume results were inconclusive, surprisingly, there was an increase in attacks after blocking Shodan scans. With the author inferring that malicious threat actors actively use Shodan for initial device discovery. However, the experiment was solely focused on blocking Shodan and did not consider other IWS platforms in circulation or other outside factors, potentially leading to misleading and unreliable results.

The process of IWS fundamentally revolves around the concept of network scanning. Employing two scanning techniques known as passive and active scanning, which, when combined, can be used to achieve full IPv4 Internet-wide scan coverage. Ultimately, both techniques come with their advantages and shortcomings, but to yield the best scan results, a combination of both techniques should be executed efficiently (Sherry, 2020). An illustration highlighting the differences between the two scanning techniques can be found in Figure 2 over the page.

Passive scanning techniques are non-intrusive and are performed by analysing network packets passing through an observation point, often comprised of a barebones system with comprehensive network monitoring and logging software installed (Xu *et al.,* 2011). This network snooping technique's very nature makes it nearly impossible for either of the parties involved in the communication to detect it. However, this technique is ineffective against targets with no active network activity to monitor, as services running with no activity - not receiving or transmitting any data – will not be detected by passive scanning (Bartlett, Heidemann, and Papadopoulos, 2007). Another drawback is that the observation point needs to be located somewhere along the network transmitting the data, which in most scenarios is an infeasible solution (Bou-Harb, Debbabi, and Assi, 2014).

Active scanning techniques directly communicate with target systems to identify and gather information. Although a lot more reliable than passive scanning, these scans are more intrusive as an active connection to the target is necessary. This technique can be performed over a wide range of protocols, using varying

approaches, which include but are not limited to ICMP, Xmas scan, as well as both five TCP-based and two UDP-based techniques (Bano *et al.,* 2018). The nature of directly connecting with the target allows for more detailed data to be obtained than when conducting passive scanning. Common IWS platform features such as service banner grabbing – allowing for version detection of running services to be identified - are made possible because of active scanning. Moreover, unlike passive scanning, there is no requirement for the strategic placement of observation devices to monitor network traffic, and services can be identified without the need for external network activity. However, this comes with drawbacks. The scans are highly detectable - non-stealth - and firewalls frequently block benign active scanners if they are deemed too intrusive, creating inconsistencies in results. Recent research into the impact active scanning can have on industrial network devices found that active scanning still has a detrimental effect on the performance of low-powered devices (Hanka *et al*., 2021). Industrial network devices are not alone. The impact is much more widespread, with researchers from Tohoku University in Japan finding that mass port scanning can cause network congestion on wireless general consumer IoT devices, which can negatively affect scan results and devices (Hashida, Kawamoto and Kato, 2020). The researchers investigated the relationship between IoT data throughput and mass port scanning speed to propose a novel analytical model for analysing the impact of scanning traffic on wireless networks. Another major downfall to active scanning is that data obtained is quickly out of date, requiring additional intrusive scans to pick up changes (Deraison and Gula, 2009).

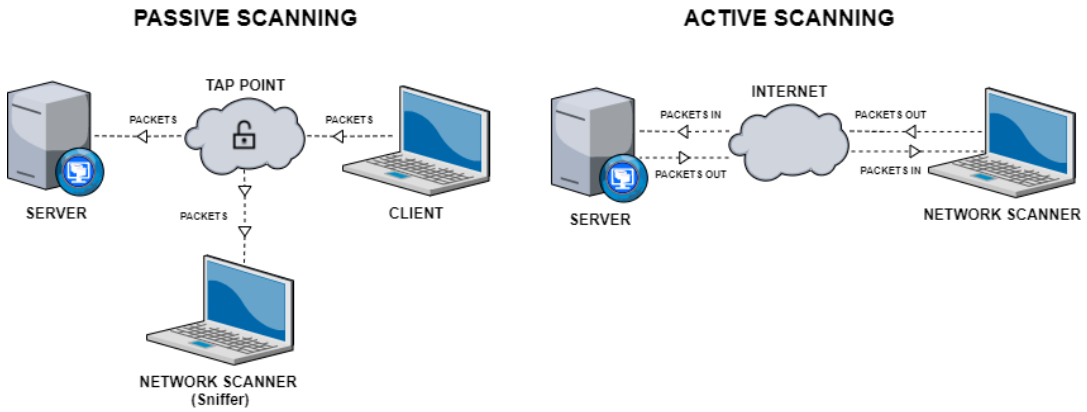


Figure 2: Passive and Active Scanning

These two scanning techniques in Figure 2 form the fundamental concept behind IWS. However, due to the strategical complexity and related ethical constraints of placing multiple observation points, passive scanning is not a singular viable solution for conducting efficient IWS. Consequently, prevalent publicly available IWS research tools and platforms primarily adopt active scanning techniques, making IWS activity a form of active reconnaissance by its very nature. Active scanning is a type of reconnaissance that is traditionally associated with malicious activity, particularly sophisticated worms and botnets (Bou-Harb, Debbabi, and Assi, 2014). The Mirai's botnet is a prime example of such. Efficiently utilising IWS techniques to spread rampantly amongst vulnerable devices, successfully compromising and controlling nearly 65,000 devices within the first 20 hours of deployment to launch large scale Distributed Denial-of-Service (DDoS) attacks (Antonakakis *et al.*, 2017). Another botnet, known as the Carna botnet, targeted devices with default telnet credentials using various methods, including ICMP ping, reverse DNS, and SYN scans, to continuously infect devices (Anonymous, 2012). Results obtained from Carna, widely known as the 'worlds first *'nice'* botnet', formed the Internet Census 2012, a report containing information on over 1.3

billion addresses in the IPv4 namespace (McMillan, 2013). This was the first time the concept of IWS gained traction as a viable methodology for security researchers. This influenced an influx in the development and deployment of IWS tooling, such as ZMap (2020) and Masscan (2021), which both released in 2013 with the goal of scanning the full IPv4 namespace. Durumeric, Bailey, and Halderman (2014) found evidence that most Internet-wide scans were being conducted by academic researchers. Drawing further attention and, more importantly, changing the persona of the way security researchers preserved IWS.

As previously discussed, Shodan may be the most well-established and researched IWS project, but it is not the only one. Other scanners found within the literature reviewed include Censys, BinaryEdge, ZoomEye (2021), Fofa (2021), and Onyphe (2020).

Censys, a tool commonly used by researchers, is based on the ZMap Project and was originally developed by a team of researchers from the University of Michigan. Censys, similar to Shodan, scans and indexes the whole of the visible IPv4 namespace and derives much of the same device attributes as Shodan. However, Censys has superior port scanning capabilities with claims of being able to scan the entirety of the internet for 2029 ports in quick succession (Censys, 2020). A claim endorsed by a SANS member, monitoring networking scan activity via honeypots over a two-month period (Guy, 2020). The platform enrols a monetised tiered access system, allowing users the ability to query a multitude of relevant device attributes to help evaluate the security of Internet-connected devices (Durumeric *et al.*, 2015). However, additional feature limited non-monetised account offerings are available to the general public and privileged researcher accounts.

BinaryEdge scans the entirety of the internet similar to the formerly cited platforms and was recently acquired by Coalition Inc, a US-based insurance firm (Motta, 2020). There is a dearth of technical research on the BinaryEdge tool, with what little is out there being focussed on the effects and effectiveness of benign Internet-wide scanners. Researchers recently conducted a survey comparing cyberspace search engines, including BinaryEdge. The results of this study indicated that BinaryEdge was a valid competitor in the cyberspace search engine market, with impressive scan results and underlying architecture (Li *et al*., 2021). Evidently so, as a security researcher, come blogger recently discovered a Thai Database that disclosed 8.3 Billion internet records thanks to the help of the BinaryEdge tool in conjunction with Shodan (xxdesmus, 2020). Another notable report using the platform surfaced in 2016, detailing how an alarming quantity of crucial devices were exposed at multiple levels, with high numbers of businesses found to be exposing terabytes of data (Morisson, Riley and Poupino, 2016).

ZoomEye, a cyberspace search engine developed and released in 2013 by Knownsec Inc, a Chinese based security company, uses their proprietary tools XMap and WMap, a port scanner and web crawler respectively, to carry out IWS. ZoomEye enables users to search for devices, services, and sites across the internet and is a popular Chinese based platform (Salame, 2020). Excluding the odd exception, the bulk of research written on the tool is in Chinese. English papers are few and far between and often evaluate the tool collectively with other cyberspace search engines. The authors Zhao *et al. (*2020) compared several aspects of ZoomEye against competing IWS platforms. Concluding that ZoomEye typically collected the most results across varying protocols compared to all platforms tested, as can be seen in Figure 3.

Devices	Vendors	Zoomeye	Shodan	Censys	Fofa	NTI
Router	TP-Link	1,047,861	248,062	261,149	924,950	743,119
IP camera	Hikvision	3,607,552	130,836	124,238	427,935	2,749,631
Printer	HP	210,507	119,183	87,343	148,309	127,625
Antminer	Bitmain	1,158	344	306	2,619	792
PACS	Dicoogle	10	1	2	1	0
ModbusGW	Modbus	47	14	8	9	36

Figure 3: IWS Scanning Detection Comparison (Zhao et al., 2020)

Fofa, another cyberspace search engine originating from China, was designed to identify Internet-connected devices to provide a real-time stream of IWS data, making it accessible to the masses (Fofa, 2021). The platform's offering of a unique vulnerability script marketplace generated from proof-of-concept (PoC) and N-day vulnerabilities ensures it stands out from competing platforms. The malicious scripts within this marketplace can be subsequently purchased by customers and run against devices in a bid to enhance their security (Zhao *et al.,* 2020). Considerable amounts of available related works are in Chinese. However, a related paper in English by Nabha and Sbeyti (2020), researchers from the Arab Open University, produced a case study investigating how vulnerabilities in MRI scanners could be exploited. Their methodology indicated that Fofa would be an ideal platform for gathering information on the network perimeters of such devices (Nabha and Sbeyti, 2020). However, this conclusion was reached with a substantial lack of evidence backing up the suggestion of the platform. Additionally, the results in Figure 3 show Fofa's impressive scanning reach.

Onyphe is a French-based platform branded as a cyber defence search engine, gathering information on Internet-connected devices via various means, primarily by listening for internet background noise and performing active web crawling (Onyphe, 2020). Founded in 2017, Onyphe is a relatively young entry to the IWS space. Therefore, a lack of academic literature on the tool is to be expected. However, several researchers have detected Onyphe probes across the internet, hinting at its widespread scanning capabilities. Ferretti, Pogliani, and Zanero (2019) set out to find the level of interest scanners have in industrial control system (ICS) devices by setting up a set of low-profile honeypots distributed around the globe. Their findings revealed the frequency of scans by scan origin on low-level programmable logic controllers (PLC) services. From the gathered statistics, a reasonable sized proportion of scans can be linked to Onyphe probes, indicating that Onyphe has a respectable scanning reach across various protocols.

There are a plethora of other accessible IWS projects available (Borbolla, 2019). Despite this, due to the abundance of literature, vast quantities of documentation, and sophisticated features, four of the previously discussed IWS platforms - Shodan, Censys, BinaryEdge, and Onyphe – are the only platforms selected as appropriate for further investigation. These aspects contrasts the two aforementioned Chinese based competitors, Fofa and ZoomEye, which have been found to lack sufficient English based research surrounding their capabilities. Additionally, there are known cases of businesses singling out China, blocking large associated network ranges, which could have an adverse effect on Chinese-based scanners' overall reliability (Matherly, 2014). Also, the ongoing tension between China and western democracy, due to continuing political turmoil, has the potential to escalate exponentially, having the

capacity to negatively influence these Chinese-based platforms. On the other hand, Shodan and Censys both have a long history of research backing them, stretching to their founding (Schearer, 2010; Durmeric *et al.,* 2015). BinaryEdge and Onyphe lack the same extent of research available, but available research is creditable and highlights the potential of both platforms.

2.2.1 Selected Internet-wide Scanner Platform Comparisons

This section will critically evaluate academic literature and other creditable sources to compare the previously highlighted four IWS projects of interest - Shodan, Censys, BinaryEdge and Onyphe - in line with the key research questions. For this review, the vulnerability reporting functionality if present on each platform, will not be considered as previous research has already deemed the features to be basic and limited in nature (O'Hare, Macfarlane and Lo, 2019). Only the previously drawn upon IWS platforms were considered for further review in this section, consisting of an in-depth analysis specifically evaluating each platform on the following criteria:

- Accounts and Features
- Scanning Capabilities
- Application Programming Interface and Documentation

2.2.1.1 Accounts and Features

The fundamental purpose of IWS platforms is to scan the entirety of the internet periodically, a process that takes time and often an immense amount of distributed computing power, which ultimately costs the platform operator financially. For this reason, all the selected platforms are legally registered entities, offering differing takes on a monetised account system based on user usage. Additionally, all offer free tier-level accounts with varying degrees of limited functionality. Such free accounts are of most interest to keep the project in scope, using freely public accessible data sources.

Shodan offers a variety of features accessible via a user-friendly web interface which is showcased in Figure 4. Non-registered users can utilise services via the web interface. However, access to data and specific features is heavily limited to prevent abuse (Shodan, 2021). The platform's main paid account model is primarily based on a four-tier system which includes the following tiers: Freelancer, Small Business, Corporate and Enterprise Data License. The pricing of all tier levels is billed monthly, with the price incrementing according to the selected tier level, starting at $59 per month for the Freelancer account. Limitations decrease, and features increase as individuals progress up the account hierarchy, with features such as advanced filters, paging, and network monitoring only available to academic and paid account holders. Once a paid subscription is obtained, accounts are allocated credits which are renewed monthly. Two categories of credits exist, with previous depreciated credit systems containing more categories. The first category of credits are query credits, with a creditable query in Shodan being classified as either using filters or navigating beyond the first page of results (Matherly, 2017). The other type of credits available are known as scan credits and can be used to force Shodan to conduct an on-demand scan on a given target; these are used up at a rate of one credit per target (Shodan, 2020). In addition to the monthly account options, a further non-monthly account option known as member is available for a one-off payment of $49 and gives users 100 query and scan credits to use each month.

Furthermore, all Shodan plans come with Application Programming Interface (API) access, and the usage amount is based upon the aforementioned credit system (Matherly, 2017).

Notably, Shodan limits access to the vulnerability search filter until at least the small business tier priced at $299 per month (Shodan, 2021a). The feature is exceptionally powerful, allowing users to search the whole of the platform's dataset by vulnerability. This project aims to reproduce a similar feature locally, based on data already retrieved from Shodan.

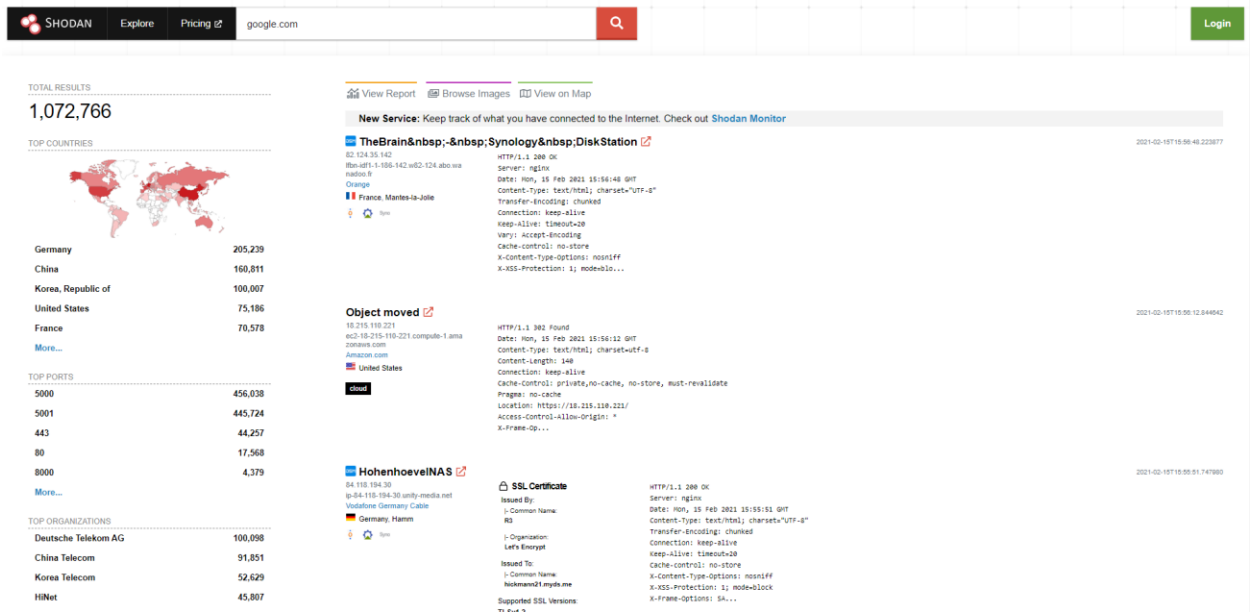


Figure 4: Shodan Search via Web Interface

Censys provides a clean, aesthetically appealing web interface that mirrors many of the same attributes as Shodan, as can be observed in Figure 5. Censys users can run a finite number of queries without registration, but this feature is heavily limited to prevent abuse (Censys, 2021). The Censys platform is plan-based and has two separate account lines besides their free plan, Censys Basic and Censys Enterprise. Registration is mandatory for all plans. The free plan allows access to both the API and web interface and includes all features under the basic line plans, but with a limit of 250 queries monthly and a maximum of 1000 results per query. A pleasant difference in comparisons to the Shodan tiered system, which has stricter limits on both features and scan quantity.

Further plans can be purchased under the basic line for an undisclosed monthly sum by contacting their sales team; such plans offer a significantly higher quantity of queries per month, results per query, and the ability for multiuser access. The Censys enterprise line offers a vast array of additional scan data in addition to Google BigQuery access and a raw JSON data feed (Censys, 2021a). As a result of Censys originating off the back of a university research project, it has remained committed to providing data to the research community at no extra cost, offering privileged data access to researchers (Censys, 2021b). Another evident advantage Censys boasts over the other platforms considered in this review is the sheer footprint of data available, with enterprise customers able to access IWS data containing over 2029 frequently scanned ports (Censys, 2021c).

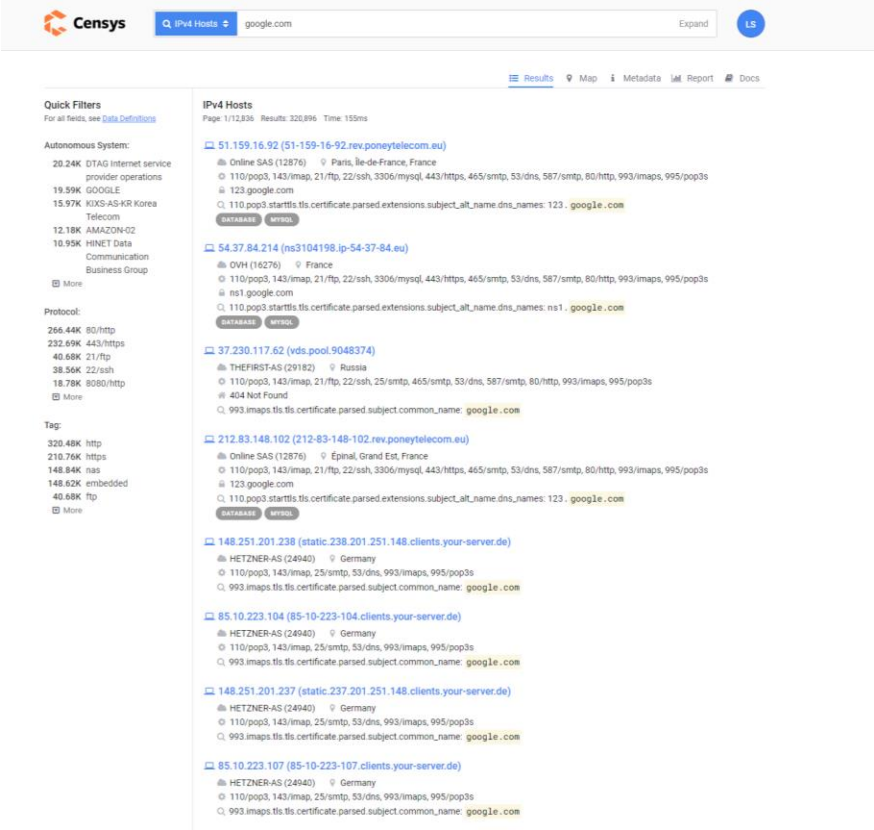


Figure 5: Censys Search via Web Interface

BinaryEdge, known initially as the 40fy platform, focuses on acquiring, analysing and labelling IWS data. The platform can be accessed via either its sleek modern web interface or its series of robust APIs (BinaryEdge, 2021a). The platform overall is well rounded and has a few unique offerings over Shodan, Censys and Onyphe. These unique features include access to dataleak, torrent, risk scoring and honeypot sensor data (BinaryEdge, 2021b). Access to the BinaryEdge platform requires an account, with no option for unregistered users to utilise the platform. A slight drawback over the other platforms being reviewed. A free account option is provided, allowing for access to the web and API interfaces to be granted, with a 250 query per month limit. However, similarly to Shodan and unlike Censys, this platform significantly limits features based on the plan selected (BinaryEdge, 2021c). With key features such as dataleaks and torrent detection requiring paid plans. A monthly pricing scheme is in place, which consists of four plans: Free, Starter, Business and Enterprise. The higher the plan, the greater the access to features and monthly query allowance. In Figure 6, the BinaryEdge web interface can be observed.

The focused approach BinaryEdge takes to gather, process and correlate data from various sources sets it apart from other IWS platforms. With an overall capacity of scanning 200 ports per month, the platform brings a truly unique offering to the table with its vast array of unique features.

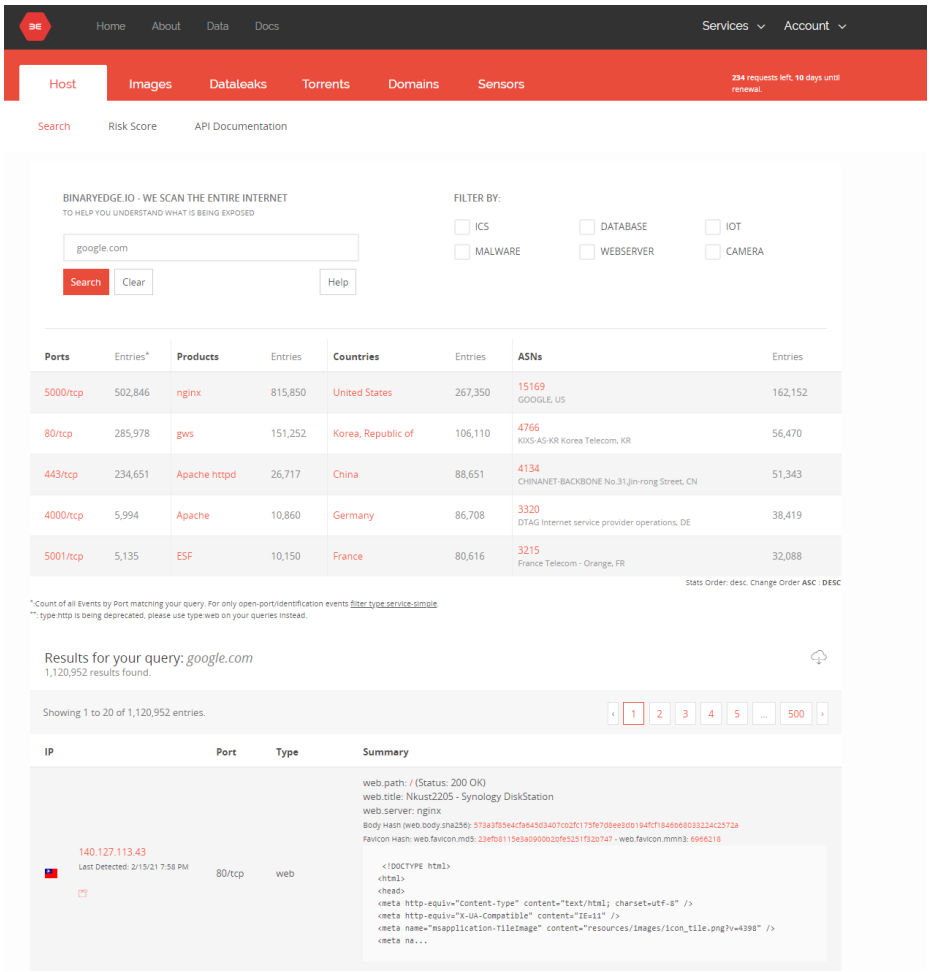


Figure 6: BinaryEdge Search via Web Interface

Onyphe access can be achieved via a simplistic web interface which can be found in Figure 7, in addition to being reachable via an API. Similarly, to Censys a two-phase plan-based billing system is in place to limit user access. This consists of two lines, Individuals and Enterprises, the latter of which is billed monthly. Fortunately, no account is required to utilise the standard web front-end search feature provided by Onyphe. However, a daily limitation on the number of searches is enforced to prevent abuse (Onyphe, 2020). Additionally, it must be pointed out that the standard search only accepts a minimal variety of search parameters (Onyphe, 2020a). API access requires an account that can be created for free. However, queries are limited to a maximum of 2,500 results per month or 250 queries, whatever comes first, and search filters are heavily limited. Paid plans can be purchased to reduce filter restrictions further, increase query results per month and increase access to more advanced features such as on-demand scanning.

Out of the reviewed scanners, Onyphe scans the least amount of ports monthly at only 100 (Onyphe, 2020). However, it correlates data from several unique sources that other platforms appear to have overlooked, including multiple IP threat lists and popular paste sites.

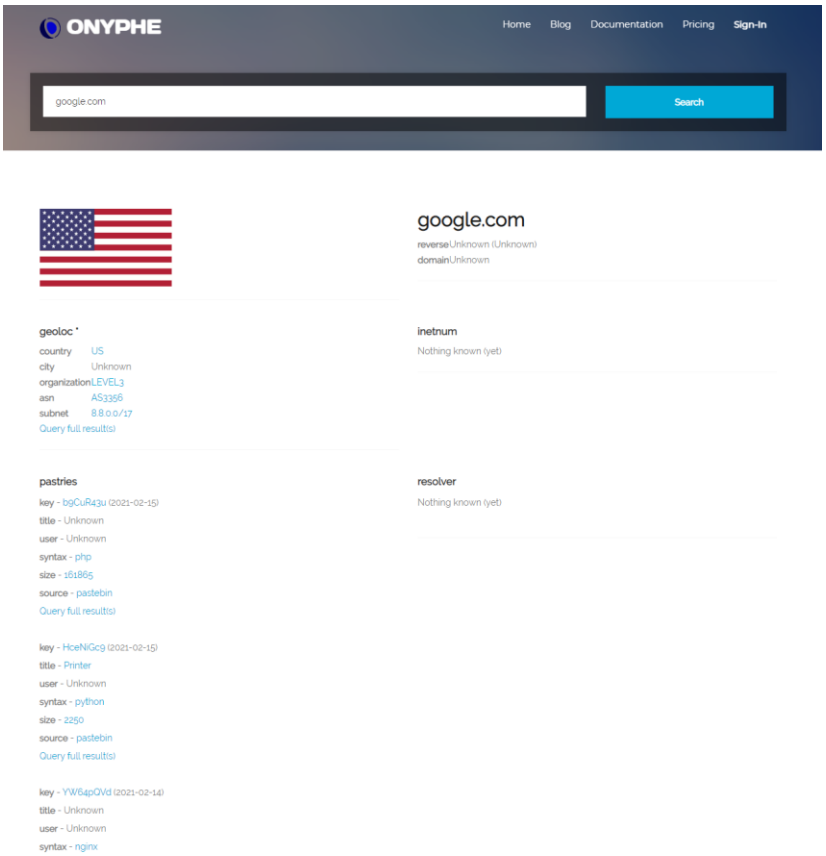


Figure 7: Onyphe Search via Web Interface

A comparison table was constructed to consolidate all previous restrictions and limitations in place across free accounts on the four considered IWS platforms under review. The table looks explicitly at comparing the difference between having a free account as opposed to not; this comparison is shown in Table 1. For the purpose of consistency, max query result limits were derived under the assumption of no filters, just a raw singular IP lookup.

Table 1: Comparison of IWS Platform - No Account and Free Account Limitations

	No Account					
PLATFORM	**Web Interface Access**	**API Access**	**Search Filters (Web & API)**	**Daily Web Restrictions**	**API Restrictions**	**Max Query Result (Web)**
Shodan	Yes	No	Limited	10 Queries	None	10 Results
Censys	Yes	No	Extremely Lenient	10 Queries	None	10 Results
BinaryEdge	No	No	None	None	None	None
Onyphe	Yes	No	Limited	10 Queries	None	10 Results
	Free Account					
PLATFORM	**Web Interface Access**	**API Access**	**Search Filters (Web & API)**	**Monthly Query Limit (Web)**	**API Restrictions**	**Max Query Result (Web & API)**
Shodan	Yes	Yes	Limited	∞ Queries	50 Results	∞

Censys	Yes	Yes	Extremely Lenient	250 Queries*	250 Queries *	1,000 Results
BinaryEdge	Yes	Yes	Lenient	250 Queries*	250 Queries *	1,000 Results
Onyphe	Yes	Yes	Limited	250 Queries*	250 Queries *	1,000 Results

** Web and API query limits are interlinked and will decrease in tandem with each other.*

From Table 1, it is clear that of the four IWS platforms analysed, Shodan and Censys are arguably the most lenient in terms of search filters and quotas. This result was expected because of the well-established nature and extent of past surrounding literature as well as the research-orientated nature of the platforms; all factors that are great incentives to draw in new clientele which further compounds the growth of the platforms. BinaryEdge was found as the only platform to offer no support for unregistered users, and Onyphe was observed as having the most restrictive free account access tier. However, overall, the platforms all have similar strict limitations for free users, especially when it comes to API access, as is expected in an attempt to prevent abuse. Shodan and Onyphe both were the only platforms reviewed to disallow CIDR scanning on their free tiers. However, although an academic account on Shodan can avert this issue, Onyphe has no equivalent free work around, which could pose restrictions on the use of this platform within this project during the implementation stage.

2.2.1.2 Scanning Capabilities

Each platform has several cyber scanning approaches which it can implement to achieve full IPv4 IWS capabilities. Bou-Harb, Debbabi, and Assi (2014) describe in-depth the underlying techniques of wide range, target-specific, single source and distributed scanning methodologies. From this paper, it can be inferred that IWS platforms primarily adopt a mixture of techniques under the categories of single source and distributed scanning. An illustration of single source horizontal scanning and distributed vertical and horizontal scanning can be seen in Figure 8.

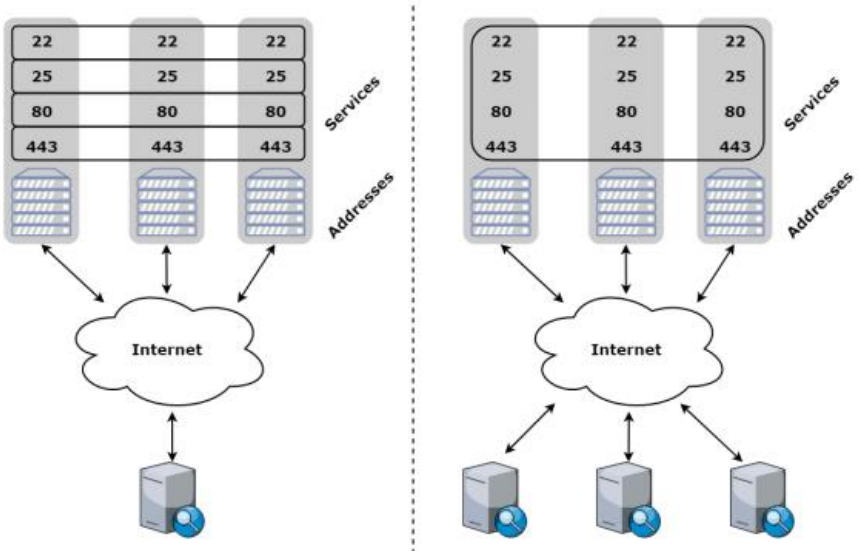


Figure 8: Single Source and Distributed Scanning Comparison (O'Hare, Macfarlane and Lo, 2019)

The Shodan platform adopts the distributed scanning methodology, scanning both horizontal and vertical with scans running continuously across multiple locations distributed worldwide, enabling the platform to gather detailed port and service information on identified devices (Matherly, 2016). This approach of using multiple locations lessens the probability of scans getting blocked by targets. However, this added complexity has been found by researchers in the past to cause noticeable delays in results, potentially rendering data obtained via Shodan prone to being outdated and therefore incorrect (Bodenheim *et al.,* 2014). The Shodan founder, John Matherly, has previously stated that he is aware that his approach is not the fastest available but favours the scalability that it provides (Matherly, 2014). Additional recent research has emerged explicitly investigating how the geographic origin of scans can negatively affect results produced by Internet-wide scanners. They experimented using seven different scan origins and discovered that results from each varied widely. They found that coverage by a single scan origin missed more than twice the number of HTTP and HTTPS services initially expected – with the expectation being based upon official ZMAP paper (Durumeric, Wustrow, and Halderman, 2013). The researchers further investigated and concluded a multitude of reasons behind the lack of consistency across scan origins, such as the blocking of entire origins. Notably, it was found that three large hosts block Censys scans; these hosts make up for 4% of all total hosts accountable. Highlighting that the scan origin is an essential factor to consider (Wan *et al.,* 2020). The key factors found to affect the accuracy of results negatively were IP reputation and previous scanning behaviour, emphasising the importance of using multiple scan origins to obtain reliable data.

A recent paper surveying several established cyber search engines, including Shodan and Censys, concluded via experimentation that Censys had superior scanning frequency compared to Shodan, as is highlighted in Table 2.

Table 2: Comparison of Scanning Frequencies (Li et al., 2021)

Protocol (port)	Shodan	Censys
HTTP (80/TCP)	10 days	2 days
TELNET (23/TCP)	24 days	2 days
HTTPS (443/TCP)	9 days	1 day
FTP (21/TCP)	13 days	2 days
SSH (22/TCP)	10 days	3 days

Analysing the results in Table 2, Censys can be seen to have around a five times better scanning frequency than Shodan across the ports selected, theoretically making its data more reliable and up to date. The Censys platform, as mentioned previously, is based on the ZMap IWS project, which adopts a different approach to scanning than Shodan and solely performs single source horizontal scanning on targets. Scanning frequencies differ by service, following strict daily, biweekly and weekly schedules, with each full Internet-wide scan running over a 24-hour time period upon being initialised (Durumeric *et al.,* 2015). As a result of Censys scans being more frequent, as shown in Table 2, the data obtained from them is more up to date and relevant. Although, due to the scanning methodology of choice, it can be argued that Censys scans are inherently noisier than their Shodan counterpart, resulting in a higher probability of scans getting blocked by security measures enrolled on target networks.

In contrast to Shodan and Censys, BinaryEdge and Onyphe have an absence of publicly available literature surrounding their backend infrastructure and scanning methodologies. Although specific technical details

around the two platforms is an area that requires more research, it is known that BinaryEdge utilises a powerful closed source scanner that adopts a similar distributed scanning methodology to Shodan (Henriques, 2015). Researchers have unveiled that the BinaryEdge platform has a vast disrupted network of 253 probes compared to Shodan with 27 probes (Rashid, 2018). This increases the likelihood that results from BinaryEdge will be more recent than Shodan. However, this is speculative and more research in this area is required to come to a solid consensus. The Onyphe platform has not released any details other than that they use in-house tooling to conduct IWS (Onyphe, 2019). However, an educated assumption into which scanning methodology used can be formed by using the capabilities provided by the GreyNoise (2021) platform, which analyses and classifies the background noise of the internet. GreyNoise indicates that Onyphe has 193 scanning probes situated across three continents. The associated GreyNoise results can be found in Appendix A. Furthermore, GreyNoise has indicated it has high confidence that the detected Onyphe probes are genuine (Morris, 2020), meaning that it can confidently be concluded that Onyphe is likely utilising distributed scanning techniques, similar to Shodan and BinaryEdge.

To gather a conclusive real-world comparison of the detection capabilities of Shodan, Censys, BinaryEdge and Onyphe, identical searches across all four platforms were performed in parallel to reveal any difference in device detection rates. The search queries were restricted to one UK-based educational institution to rule out any potential personal bias towards specific networks. Additionally, the constant static geographic location of the target allows for a fair testing ground, with the only changing factor being the IWS platform. Results from this comparison can be observed in Figure 9.

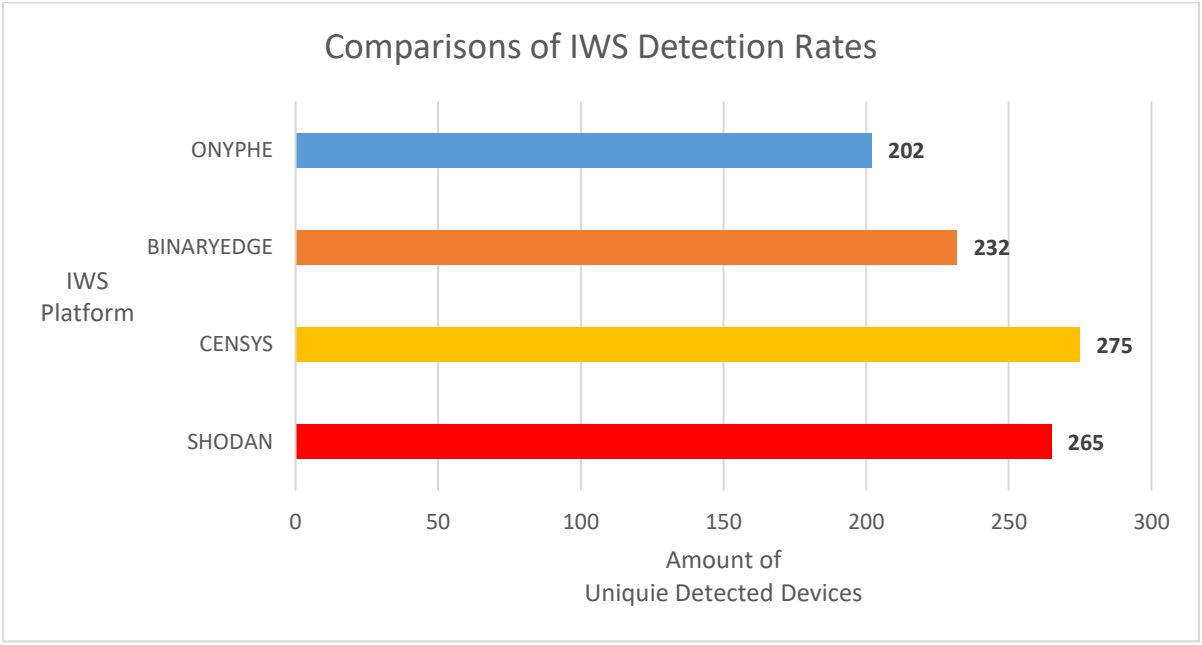


Figure 9: Comparison of IWS Detection Rates

Figure 9 indicates that Censys could detect more unique devices – IPs - than Onyphe, Shodan and BinaryEdge. This brief investigation highlights inconsistency in results between IWS platforms querying the same network range. From these results, the shortcomings of solely relying on one IWS platform for gathering information is apparent, with the varying array of factors already previously discussed having the ability to influence the results. Of these various reasons, in this case, it can be hypothesised that

network-based security systems such as intrusion prevention systems are likely to be the underlying culprit due to unexpected significant differences found between the results. Therefore, it is fair to assume that the proposed idea of correlating data from multiple Internet-wide scanners would help mitigate this problem to some degree.

The investigation results are further broken down into six common protocols known to be supported by all scanners, as can be seen in Table 3. From this breakdown, it oddly can be observed that Censys struggles to grab SMTP services. Reasons for such results are inconclusive. Additionally, BinaryEdge in this instance appears to fail to pick up on services operating over port 8080, a common port which SecurityTrails have listed as being one of the 20 most scanned ports in the cybersecurity field (Borges, 2019). The outcome is another unexpected result as the other platforms under review each successfully detect one IP address running a service on port 8080. A controversial result as BinaryEdge appears to scan this common port, as is evident in their internet Security Exposure 2016 report (Morisson, Riley and Poupino, 2016). Additionally, other devices on separate networks operating services on port 8080 appear to be showing via BinaryEdge, bringing further confusion as to why the port has not been picked up in this instance. More research in this area is needed to come to a conclusive answer.

Table 3: IWS Platform Comparison - Protocol Breakdown

PROTOCOL	SHODAN	CENSYS	BINARYEDGE	ONYPHE
25/SMTP	4	2	4	2
22/SSH	40	45	44	34
80/HTTP	174	146	172	150
443/HTTPS	160	155	172	150
53/DNS	13	27	26	27
8080/HTTP	1	1	0	1

In addition to the highlighted inconsistencies in detection rates across platforms, a table consolidating key aspects of each IWS platform's scanning capability was produced, as shown in Table 4. Fields marked with '-' are inconclusive, with no publicly available data surrounding the matter.

Table 4: IWS Platform Comparison - Key Scanning Capabilities

Feature	Shodan	Censys	BinaryEdge	Onyphe
Total amount of ports scanned	1200+	2029	200	100
Amount of ports that include banner grabbing	-	2029	-	-
Scan frequency	Medium	High	Medium	-
On-demand scanning	Yes	Yes	Yes	Yes

2.2.1.3 Application Programming Interface and Documentation

As already concluded from the accounts and features review, all four of the IWS platforms (Shodan, Censys, BinaryEdge and Onyphe) have official supported APIs which are freely accessible upon registration of a free account. Although already brought to light, all platforms understandably have strict limits on their free API usage tier to prevent abuse.

In addition to each platform providing REST APIs for data access, each has its own official Python API wrapper, making communication with the API a simplistic task. Shodan, Censys and BinaryEdge provide detailed English documentation on how to leverage their respective system's full power, including in-depth guides on how to interface with their respective REST APIs (Shodan, 2021d; Censys, 2021d; BinaryEdge, 2021a). Onyphe documentation, although in English and still helpful, is slightly lacking in-depth, which can be associated with the platform being less established in nature. This could prove to cause difficulties during the implementation stage of this project.

With Shodan being such a well-established platform, an extensive list of additional addons have been developed for Shodan by both the community and the businesses internal development team. Such extras include Shodan 2000 (Shodan, 2021b), an in-house retro style addition to the platform and various community-created dataset queries as well as a visually appealing ICS radar (Shodan, 2021c). This is by no means an exhaustive list, with many more projects out there primarily focused on expanding the use case of the Shodan platform. No comparable internal or community-driven projects could be found for either of the other IWS platforms reviewed in-depth, other than repositories of individuals interacting with platform APIs. Shodan also provides various clients and libraries supporting various other programming languages outside of Python to interact with the platform.

2.3 Current Vulnerability Assessment Tooling

Data correlation between known-vulnerability databases is a well-practised and researched approach adopted by many IWS platforms to report basic service vulnerability information (Durumeric *et al.,* 2014; Simon, Moucha and Keller, 2017). However, when it comes to traditional dedicated vulnerability assessment tooling, more intrusive active scanning approaches are often executed. This section will investigate and compare the current state of the vulnerability assessment tooling market.

The numerous free and commercial dedicated tools developed to assess networks for vulnerable devices use a combination of both active and passive techniques to assess targets. Few dedicated tools conduct assessments solely passively (Genge and Enăchescu, 2016; Genege, Haller, & Enăchescu, 2016). Nessus (2021) and Tripwire IP360 (2019) are both reputable commercial industry-leading tools in the network vulnerability assessment area, making heavy use of active scanning techniques to obtain results. Both work by systematically profiling assets within a given network range before attempting to identify vulnerabilities within detected assets. Although accurate, this has been found to cause performance and stability issues on targeted devices, especially on low-powered devices (Hanka *et al.*, 2021). Making these tools not practical in particular situations, such as in networks with high numbers of low powered mission-critical devices, a common scenario within the ICS sector. OpenVAS (2020) is another well-established tool in this field that has the advantage of being open-source, but unfortunately suffers from the same drawbacks that Nessus and the other previously mentioned tools brings due to a heavy reliance on active scanning.

Entirely passive tools exist, although functionality often tends to be lacking compared to tools adopting active scanning techniques. P0f and PRADS fall under the category of passive traffic fingerprinting tools (Zalewski, 2014; Fjellskål, 2020). Such tools allow for efficient passive analysis of network traffic, enabling the identification of known services amongst a host of other related information, particularly operating

system and IP address information (Geneg, Haller and Enăchescu, 2016). Unfortunately, neither of these tools further analyses the retrieved service data to automate the assignment of vulnerabilities. NetGlean is a distrusted network security scanner that uses both active scanning and passive packet analysis to audit the security of complex networks (Manes *et al., 2005*). This is done by deploying a range of sensors distributed network-wide, which individually fingerprint systems and correlate the data back to a central point.

In addition to the variety of passive and active tools above, a finite amount of CAR based security scanners has been found within the literature. The Shodan Vulnerability Assessment Tool (ShoVAT) was developed by the authors Genge and Enăchescu (2016) to practically implement the novel idea of passively identifying vulnerabilities using IWS platforms. At the time, it was the only published work located to rely solely on IWS platforms to achieve this. The ShoVAT tool achieves its goal by ingesting data from Shodan and then generating CPEs, which are then associated with known-vulnerability data sourced from the National Vulnerability Database (NVD) (Genge and Enăchescu, 2016). This tool, and surrounding research, has received criticism for being too focused on the performance aspect of the tool as opposed to the accuracy (Simon, Moucha and Keller, 2017). This research was further refined by O'Hare, Macfarlane and Lo (2019), resulting in Scout, a CAR known-vulnerability assessment tool that prioritises the accuracy of automated vulnerability assignment over performance and ingests data from Censys rather than Shodan. Another researcher Rodrigues (2019), developed a tool correlating Shodan generated CPEs with vulnerability data, in addition to extensive data enrichment via a multitude of data sources. Consequently, this approach to CAR vulnerability assessment is arguably the most straightforward but, at the same time, the most unreliable due to the sole dependence on Shodan for both device and service CPE data. The tool developed by Rodrigues (2019) spans far beyond the scope of the ShoVAT and Scout tools, falling more into the realm of a fully functioning OSINT platform, similar to Spiderfoot (2021).

ShoVAT, Scout and the solution developed by Rodrigues (2019) are the only CAR known-vulnerability assessment tools within the literature reviewed. All have their differences; however, they all primarily correlate known-vulnerability data from one centralised source, the NVD. A publicly available vulnerability database project managed by MITRE and backed by the National Institute of Standards and Technology (NIST, 2018). The database consists of Common Vulnerabilities and Exposures (CVE), which aim to identify and define known vulnerabilities using various numerical and text metrics. Information within each CVE includes attributes such as summary, severity via the Common Vulnerability Score System (CVSS) and vulnerable configurations through CPEs. A consolidated simplified version of available CVE attributes can be found in Figure 10.

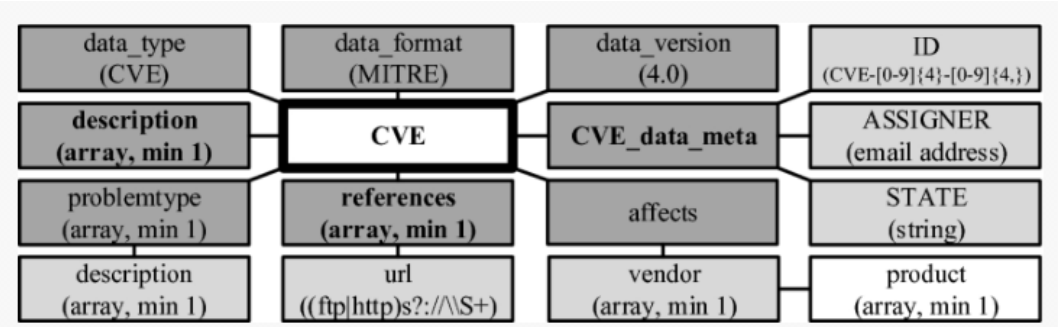


Figure 10: Simplified CVE Attributes (Schlette et al., 2020)

The NVD is widely valued by the ill-informed as a credible source that follows the Security Content Automation Protocol (SCAP), which is specifically developed to ensure the standardisation of security flaws within communication and documentation (Waltermire *et al.,* 2018). However, even with this measure in place, the NVD has been found to be littered with flaws. Multiple researchers have pointed out an abundance of issues over the years, ranging from the poor readability of data to the mediocre accuracy of the data itself (Khadilkar, Rachapalli, and Thuraisingham, 2010). Recent research conducted by authors from Tianjin University and the Australian National University set out to tackle common problems associated with the NVD, explicitly looking for solutions to the considerable amount of missing information in the database. The researchers designed a Natural Language Processing (NLP) model based upon data pulled from the summary component of CVEs contained within the NVD. Allowing them to effectively fill in missing gaps within the original dataset by achieving data taxonomy (Guo, Xing and Li, 2020). Overall, the researchers' machine learning model successfully achieved an F-measure of 0.780 with a precision of 0.778 and a recall of 0.799.

Furthermore, another critical issue highlighted by the author Rodriguez *et al.* (2017) indicated that the timeliness between vulnerabilities being disclosed and appearing on the NVD in contrast to alternative vulnerability disclosure means was significant, as shown in Figure 11.

Source	Total number of vulnerabilities collected from 2017	Total number of NVD delays	Highest disclosure delay from NVD	
SecurityFocus	5525	3973	CVE-2017-5637	244 days
ExploitDB	732	263	CVE-2017-1002000	195 days
Cisco	321	291	CVE-2017-3848	37 days
Wireshark	50	50	CVE-2017-6467	392 days
Microsoft	590	406	CVE-2017-8575	10 days

Figure 11: Disclosure Delays of NVD (Rodriguez et al., 2018)

The results from Figure 11 shine light on the significant lag in the reporting of vulnerabilities to the NVD, an issue emphasised time and time again by researchers (Rodriguez *et al.,* 2018; Leverett, Rhode and Wedgbury, 2020). Examining a more recent report revealed four of the top ten vulnerabilities identified in the 2020 Open Source Security and Risk Analysis report by Synopsys (2020), were found not to be directly associated with CVEs at the time of publication. Additionally, the NVD was identified as having an average vulnerability reporting delay of 27 days (Synopsys, 2020). As a result, these facts evidently show that the timeliness and lag of vulnerability reports being published on the NVD is still a present issue.

Contrary to the above research, Anwar *et al.* (2020) investigated recent improvements to the NVD, finding that the timeliness of CVE publication, also known as the lag time, has improved over time. With the lag of low severity vulnerabilities improving by 37%, medium by 41% and high by 65% when comparing v2 labels with v1 labels. Subsequently, another related piece of work has explored how this time lag could be further minimised by automating the CPE creation process, which is traditionally lengthy and manual. The researchers propose using a machine learning technique known as Named Entity Recognition (NER) to automatically and accurately construct CPEs, based upon the contents within the summaries of CVEs (Wåreus and Hell, 2020). The proposed solution achieves an F-measure of 0.86 with a precision of 0.857 and recall of 0.865. Overall, the study proved to be a success, offering a valid method for the NVD to improve the efficiency of its operations and hence decrease the overall lag time of reported

vulnerabilities. However, although it is known that the NVD is investigating this technique of automating the CPE creation process, it is not publicly known if such a solution has been adopted.

A comparison of the active and passive scanning nature of the previously mentioned tools derived from reviewed literature can be observed within Table 5, where '●' is used to denote a weak or no dependence, '●●' is used to denote a medium dependence and '●●● ' is used to denote a strong dependence on scanning techniques.

Table 5: Comparison of Tool Scanning Techniques

Tool	Passive	Active
Nessus	●	●●●
Tripwire IP360	●●	●●●
OpenVAS	●	●●●
P0f	●●●	●
PRADS	●●●	●
NetGlean	●	●●●
ShoVAT	●●●	●
Scout	●●●	●

2.4 Service Banner Analysis

To identify known vulnerabilities using vulnerability databases, specifically the NVD, accurate data correlation between the CPEs and CVEs is vital. The CPE is a standardised format for identifying and outlining applications, hardware, and operating systems. Two current CPE specifications exist, 2.3 and 2.2, with the former being the most recent implementation (NIST, 2021). The CVE standard is a format for identifying, categorising, rating and describing publicly disclosed vulnerabilities. Both standards follow the SCAP framework (Fitzgerald and Foley, 2013). Within the NVD, each unique CVE is associated with multiple associated CPE values, each of which corresponds to either an application, piece of hardware or operating systems. This data correlation enables assets and services affected by vulnerabilities to be linked and identified easily. This section will investigate how researchers are constructing and deriving CPE values from service banners.

The latest CPE specification created by NIST outlines the CPE naming convention, providing information on creating CPEs for validation checks (Cheikes, Waltermire and Scarfone, 2011). The naming specification describes CPEs using a total of three methods known as Well Formed Name (WFN), Uniform Resource Identifier (URI) and formatted string (FS) (Schlette *et al.,* 2020). The FS CPE encoding method is utilised by the NVD and can be seen in Figure 12. Unfortunately, as with all standards, not all developers follow the specification as intended, leading to inconsistencies in the creation of CPEs.

```
CPE:2.3:{PART}:{VENDOR}:{PRODUCT}:{VERSION}:
    {UPDATE}:{EDITION}:{LANGUAGE}:{SW_EDITION}:
    {TARGET_SW}:{TARGET_HW}:{OTHER}
```

Figure 12: CPE – FS Structure Used by NVD (Schlette et al., 2020)

Sanguino and Uetz (2017), within their research into the creation of CPE values for Vulnerability Management Systems (VMS), bring to light issues faced when attempting accurate CPE creation. The evidence shows that they identified that over 100,000 CVE entries are associated with non-existing CPE values, an important issue to consider when attempting to reproduce the CPE creation process for VMS. Consequently, this issue will directly result in broken CPE to CVE correlations, causing false negatives. Another key issue identified is the incorrect spelling of many CPEs, further negatively affecting the data correlation. Thus, increasing the chance of an incorrect CPE being matched after banner analysis. Having identified these issues, the researchers proposed an efficient solution to tackle the problem of CPE creation. The proposed solution consists of three steps, generate search terms, search by product plus vendor, and sort by version. This process is illustrated in Figure 13.

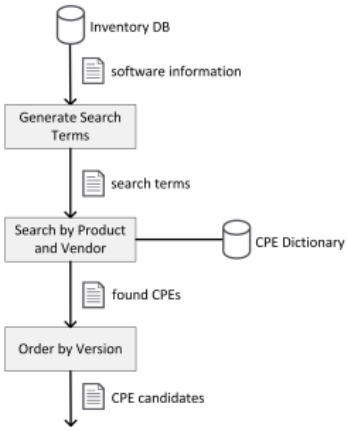


Figure 13: CPE Matching Method (Sanguino and Uetz, 2017)

In step one, WFNs are created by collecting information from GLPI databases containing software information; these are used to form search terms which in the proceeding step are used to create CPE WFN attributes. In step two, these produced CPE WFN attributes are then finally combined in accordance with the NIST CPE specification. Then thirdly, the Levenshtein distance algorithm is used to find the closest CPE match within the entire CPE dictionary. The use of this algorithm mitigates the issue of incorrectly spelt CPEs. An additional final step of order by version is essential as the previous step can return multiple close CPE matches due to the nature of the adopted Levenshtein distance algorithm approach.

The authors Na, Kim and Kim (2017) propose a method for automatically generating CPEs based solely on service banner information obtained from the Internet-wide scanners. Their proposed method involves performing keyword analysis on banner texts and comparing the results against the entire dictionary of available CPEs. This would be achieved by generating a CPE FS tree, performing keyword analysis on the banner, and then locating the closest banner relating to the keywords. In addition to this, if the OS was the only identifiable attribute from the banner analysis stage, templates of default applications installed per OS would be used, a feature referred to as basically installed software by the researchers.

The above outlined proposed solution for service banner analysis has several weaknesses. The first being that the basically installed software feature assumes that default programs have not been uninstalled, and no consideration for containerised systems is given. The fundamental banner analysis process

described also possesses a vital issue due to the heavy reliance on accurate vendor and product information needing to be extracted from service banners. It is apparent that the approach taken by the previously mentioned tool, ShoVAT, to perform the construction of CPE values from service banners leverages the power of multiple hierarchical trees using hash tables to store in-memory segments of corresponding CPE version numbers efficiently. Thus, it is avoiding the need to search through the entirety of the NVD for substrings of each CPE value (Genge and Enăchescu, 2016). The pseudocode of the algorithm implemented by ShoVAT can be seen in Figure 14. Drawbacks to such an approach revolve around updating the hash table and ensuring no incorrect or depreciated CPE values are present within the data (Sanguino and Uetz, 2017). Nevertheless, the presence of defunct CPE values would significantly increase both the false positive and false negative detection rates. However, the performance of ShoVATs method is far superior to that proposed by the authors Na, Kim and Kim (2017).

Algorithm 1: Reconstruction of CPE names from data retrieved from Shodan.

input : Shodan entry $< host, port, t, banner, os, cpe, product, version >$
output: A set of CPE names $\mathcal{R}_{cpe}$ and a set of possible OS names $\mathcal{R}_{os}$

Let V_{pat}="$I.(I.)^*I$";
$\vartheta = get\_pat(\mathsf{V}_{pat}, banner)$;
Let $\mathcal{R}_{cpe} = \{cpe\}$;
Let $\mathcal{R}_{os} = \{(os, version)\}$;
foreach v *in* ϑ **do**
    Let $\varsigma = h_v(v)$;
    if $|\varsigma| > 0$ **then**
        $\mathcal{R}_{cpe} = \mathcal{R}_{cpe} \cup \{(cpe_i, weighted\_match(cpe_i, banner))|cpe_i \in \varsigma\}$;
    else
        Let ϑ' = $get\_leaves(h'_v(get\_exact\_pattern(\mathsf{V}_{pat}, v)))$;
        Let $\varsigma' = \bigcup\{h_v(v'')|v'' \in \vartheta'\}$;
        $\mathcal{R}_{cpe} = \mathcal{R}_{cpe} \cup \{(cpe_i, weighted\_match(cpe_i, banner))|cpe_i \in \varsigma'\}$;
    end
end
$\mathcal{R}_{os} = \mathcal{R}_{os} \cup \{match\_os\_names(OS_{names}, banner)\}$;

Figure 14: ShoVAT Banner Creations Algorithm (Genge and Enăchescu, 2016)

O'Hare, Macfarlane and Lo (2019) took a blend of the previously discussed approaches to help develop the banner analysis algorithm behind Scout, with their developed approach heavily influenced by research by Sanguino and Uetz (2017). Their approach consisted of enumerating through the complete CPE dictionary provided by the NVD, searching on each iteration for a close match to service banner data returned by Censys. The selected process utilised the Levenshtein distance algorithm and appends CPE values to three lists, elected candidate, vetted candidate and accepted candidate depending on the value of symmetric difference. This process helps address issues surrounding selecting one valid CPE, as was found to be the case in previous research (Sanguino and Uetz, 2017). Further length analysis of these three lists leads to either a conclusive or inconclusive singular CPE match. An advantage over previously discussed methods. However, depreciated and inconsistent CPE values still pose a threat to this methodology, as without further validation, depreciated CPEs will result in false positives or even inconclusive results.

Building upon the literature reviewed in the previous section, an updated comparisons table of tools capable of auditing network devices for vulnerabilities can be found in Table 6. Where '●'is used to denote a weak or no dependence, '●●'is used to denote a medium dependence and '●●● 'is used to denote a strong dependence on scanning techniques.

Table 6: Further Tool Comparison

Tool	Passive	Active	Custom Banner	CPE/CVE
Nessus	●	●●●	●●	●●●
Tripwire IP360	●●	●●●	●●	●●●
OpenVAS	●	●●●	●●	●●●
P0f	●●●	●	●	●
PRADS	●●●	●	●	●
NetGlean	●	●●●	●	●
ShoVAT	●●●	●	●●	●●●
Scout	●●●	●	●●	●●●

2.5 PASSIVE DATA ENRICHMENT

The amount of data produced by Internet-connected devices around the globe at any given moment is immeasurable. However, a portion of this data is freely available and widely accessible to the public at short notice. The term open-source intelligence (OSINT) encapsulates such data accessible via publicly available sources. OSINT techniques and data are commonly adopted by intelligence services and governments in the fight against cybercrime and can be of incredible value when used correctly (Nouh *et al.,* 2019). Similarly, organisations can use OSINT to improve their overall cyber offensive and defensive capabilities by correlating multiple data sources, helping support critical cyber defence decisions and even influence proactive defence measures (Nespoli *et al.,* 2017). However, OSINT data is often overwhelming and complicated. A quote from Mathematician Clive Humby (2006) symbolises this well, stating “data is the new oil”. Much the same as oil, if data is unrefined, it is of little use. This section will examine literature in order to find OSINT sources and describe the challenges faced when working with such sources.

As powerful as OSINT is, it has several limitations. The primary factor being the quantity and complexity of data, two factors that have a detrimental effect on the ability to process the data accurately and efficiently (Fleisher, 2008). Consequently, substantial resources and sophisticated technical techniques are needed to ensure the quality and consistency of retrieved data. Another limitation is the trustworthiness of the data sources being utilised. Data should ideally be sourced from authoritative, trusted domains, but when using OSINT, this is rarely a viable option. Such sources are prone to false and inaccurate information; however, these sources also often contain copious amounts of accurate and valuable information hidden within (Pastor *et al.,* 2020). This is often the case when it comes to OSINT data. The copious amount of data requires continuous refinement to determine key important pieces of information. Finally, several ethical issues have arisen surrounding OSINT sources ability to respect the privacy of individuals. Although the data obtained is publicly available, the internet is vast and contains information not explicitly intended to be online, specifically drawing out and disclosing such information does not respect user privacy (Kandias *et al.,* 2013). Putting many OSINT data sources within a grey legal and ethical area.

A recent survey into 16 practitioners using OSINT techniques regularly determined that since the advent of GDPR - which is seen by many as being the first step in establishing societal expectations on digital

privacy - there have been few amendments to OSINT techniques used by the participating practitioners (Shere, 2020). Ultimately leaving the ethical lines still blurred, and this a legal grey area requiring further research.

Various OSINT sources and tooling can be found in literature, spanning several areas. Internet-wide scanning platforms are one such source already previously discussed. Other tools within the cybersecurity realm focus on collecting alternative information, such as DNS data, geolocations, threat actors and database dumps. This list is by no means exhaustive. A more comprehensive list can be found on the OSINT Framework website, which contains a compilation of available sources and tooling, broken down into 32 separate categories ranging from metadata to the darkweb (Nordine, 2021). Due to the broad range of sources available and fundamental need for data refinement, an abundance of OSINT tooling has emerged, consolidating a multitude of data sources and efficient processes. These factors make working with OSINT data more manageable and beneficial. Figure 15 shows an overview of the features present in a selected number of popular OSINT tools.

OSINT tool	Input				Output	Extensibility	Interface	Platform	Other feature
	Identity data	Network data	File data	Selectable data source					
FOCA	✗	Domain	File name, Folder	Google, Bing, DuckDuckGo	Identity info, Network info, File info	✗	Stand-alone program	Windows	Server discovery module
Maltego	Personal information, company, community	Domain	File URL	✗	Identity info, Network info, File info	Custom transforms	Stand-alone program	Linux, Windows, MAC	Location, Auto input/ output refeed, Results in oriented graph
Metagoofil	✗	Domain	File type	✗	Network info, File info	✗	Command line	Linux, Windows	Option to narrow results
Recon-NG	Personal information	Domain	✗	Several	Identity info, Network info, File info	✗	Command line	Linux	Location, Modules for discovery and exploitation
Shodan	Country, City, Keyword	Operating system, IP Address, Port, Host name	✗	✗	Network info	✗	Web interface	Online	Location, Webcam captures
Spiderfoot	Email, Real name, Phone Number	Domain, IP Address, Subnet, Host name	✗	Several	Network info	Custom modules	Web interface	Linux, Windows, MAC	Different types of scan, Results in oriented graph
The Harvester	Company	Domain, DNS server	✗	Several	Identity info, Network info	✗	Command line	Linux, Windows, MAC	Results in reports, Option to narrow files and results
IntelTechniques	Personal information, company, community	Domain, IP Address	File name, File type, File URL	Several	Identity info, Network info	✗	Web interface	Online	Location, Public records, OSINT virtual machine

Figure 15: Popular OSINT Tool Feature Comparison (Pastor et al., 2020)

From Figure 15, it can be observed that network and identify information are typical outputs from each tool reviewed. Upon further analysis of each tool above, a wide variety of sources were found to be adopted to produce the desired outputs, with IWS projects being found as the primary source of data. Amongst the plethora of other sources implemented by the tooling, Passive Domain Name System (PDNS) data was identified as an excellent source to be explored further. Due to its enrichment potential and ease of data correlation with IWS data. Additionally, it was identified that no tool reviewed from Figure 15 currently leveraged PDNS data to detect potential weaknesses within networks, only solely using it as a means of enrichment. Prompting a further review into PDNS.

The technique of PDNS was first described in a research paper by Florian Weimer (2005), with the aim to analyse malicious DNS traffic. The technique works by waiting on the recursive domain name server responding to requests from other domain name servers, whilst capturing and storing the traffic data in the process (Weimer, 2005). For example, users querying the DNS for a domain are first redirected to a local DNS resolver to retrieve a result. However, what if the particular result cannot be retrieved. In that case, the DNS resolver will query an external root server, followed by the corresponding authoritative name server and top-level domain server to retrieve the final result. During this process, the DNS resolver – sensor - will capture and store the requested information, which is the technique of PDNS. This process is better illustrated in Figure 16.

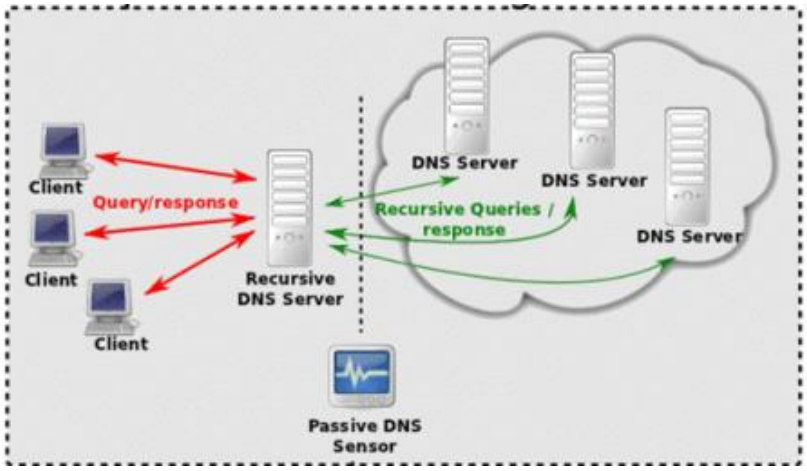


Figure 16: Passive DNS (Marchal et al., 2012)

PDNS data allows for a historical view of the global DNS and enables the functionality of many well-established research applications in the field of cybersecurity. In addition, to its original intended purpose of capturing large volumes of traffic for malicious behaviour analysis. An area in which multiple studies continue to pursue, using emerging technologies and varying analytical techniques to further improve upon the initial idea (Zdrnja, Brownlee and Wessels, 2007; Antonakakis *et al*., 2011; Bilge *et al*., 2014).

Xuanzhen, Zulie and Yuanchao (2020) discuss multiple research areas revolving around PDNS data, concluding that the primary use case of the data source is within the realm of malicious domain name detection. However, additional use cases for the data have recently emerged due to the broad adoption of machine learning surrounding automated domain classification and reputational scoring systems. This study is conclusive in highlighting several key areas in that PDNS can benefit cybersecurity. However, it missed out on its further application for detecting dangling DNS records. Such DNS records point to invalid and no longer required resources. The authors Liu, Hao and Wang (2016) find that such records are often caused by stale DNS records which are not properly purged, a common issue within large organisations that potentially leaves (sub)domains vulnerable to temporary takeover. These authors further identify that the prime cause of exploitable dangling records is down to the lack of authenticity checks on resources pointed to by DNS records.

From the above research, it is proposed that by taking advantage of historical PDNS data, it potentially could be possible to identify exploitable dangling DNS records. Identifying a gap in current research. PDNS

is just one source of OSINT explored in detail, with many more avenues out there all providing differing capabilities. However, further analysis of such sources is out of the scope of this literature review.

2.6 CONCLUSION

This chapter aimed to critically analyse and assess surrounding literature in the area of IWS, vulnerability assessment tooling, service banner analysis and passive data sources. Investigating how each of these aspects could be adopted to improve upon the effectiveness of the CAR process. The review covered each of the four previously stated focus points within the overall domain. The remainder of this section will go on to summarise and draw conclusions on each of these aspects.

An initial comprehensive review of the literature surrounding IWS techniques, tools and platforms identified several solutions currently in circulation. However, after a brief analysis of the available solutions, four IWS platforms, Shodan, Censys, BinaryEdge and Onyphe, were the only platforms selected as appropriate candidates for a further exhaustive analysis and breakdown. To be eligible for this exhaustive investigation, each platform had to meet one or more of the following criteria: an abundance of literature, vast quantities of documentation or an advanced array of features. Reviewing related direct and indirect research surrounding the four platforms led to several conclusions. Firstly, Shodan, BinaryEdge, and Onyphe were found to likely be utilising a mixture of horizontal and vertical distributed scanning techniques to scan the internet (Henriques, 2015; Matherly, 2016; GreyNoise, 2021).

In contrast, Censys was the only reviewed platform found to use just horizontal scanning methods, using the ZMap tool to perform scans (Durumeric *et al.,* 2015). This more efficient approach resulted in Censys having a better overall scanning frequency, accuracy, and having more up-to-date data than the other platforms reviewed. Comparing the accounts and limitations available on each platform found BinaryEdge and Onyphe to be the most restrictive platforms for free, and none registered users. BinaryEdge was the only platform reviewed to not be usable without registration. Censys came out on top in terms of useability for free users, with the most lenient access to filters. An evaluation of the unique IP detection rate of the considered platforms unexpectedly found each to have varying detection rates, with Censys found to have the highest detection rate. However, the reasoning behind this could not be conclusively determined, although it is thought that this is primarily down to network-based security systems blocking particular scanning probes (Shori, 2018). Further review found the four platforms to have sufficient APIs, with all apart from Onyphe providing detailed documentation. Shodan came out on top when it came to available addons, with an abundance of well-developed and maintained first and third-party attachments.

A review into current vulnerability assessment tooling identified several key solutions found in literature. For this review, tooling utilising CAR, active and passive techniques capable of vulnerability assessment were reviewed. Tools with both manual and automated vulnerability assessment processes were considered. The review found the passive scanners to lack automated vulnerability assessment processes, whilst active scanners were found to thrive in this area. However, the scanners employing active techniques were found to have a detrimental effect on network throughput (Hashida, Kawamoto and Kato, 2020). An issue not contributable with passive scanning techniques. Three tools passively leveraging actively gathered data from IWS platforms, a process known as CAR, were additionally reviewed. ShoVAT was identified as the first tool to practically implement the novel idea of passively identifying

vulnerabilities using IWS platforms and server banner analysis (Genge and Enăchescu, 2016). However, it was focused more on performance than accuracy, leading to the creation of Scout, a refined approach to the ShoVAT. The Scout tool improved on many of ShoVAT weaknesses by prioritising the accuracy of automated vulnerability assignment over performance, as well as ingesting data from the more up to date Censys platform, rather than Shodan (O'Hare, Macfarlane and Lo, 2019). Another tool by Rodrigues (2019) appears to conduct similar vulnerability assessments via the CAR process using CPE values provided by Shodan. However, there is a dearth of detailed information available surrounding this tool and its underlying processes. The adoption of the CAR process to identify known-vulnerabilities to this point has failed to utilise more than one IWS platform, potentially allowing numerous assets to go undetected. Identifying a gap in current research, that this paper looks to address.

During the review of current vulnerability assessment tooling literature, several papers indicated issues with the NVD, prompting a further investigation into the literature surrounding it. The studies reviewed indicated that the NVD suffered from numerous flaws, such as data inconsistencies, including spelling mistakes as well as missing and incorrect information (Rodriguez *et al.,* 2017). More significantly, the existence of a lengthy lag in the reporting of vulnerabilities to the NVD was found (Rodriguez *et al.*, 2017), putting systems monitored by VMS relying on this data for vulnerability detection at risk. This research leads seamlessly into reviewing service banner analysis literature, all of which was found to follow the SCAP framework by NIST (2020). Work in the field is narrow. However, of the discovered solutions for service banner analysis, the researchers Sanguino and Uetz (2017) produced the first solution, considering NVD weaknesses. Moreover, the algorithm developed for the discussed Scout tool utilised a blend of techniques from various discussed researchers and was determined to be the best solution, capable of overcoming many of the NVD weaknesses, in addition to being extremely versatile (O'Hare, Macfarlane and Lo, 2019).

Additional passive data enrichment options were reviewed to gain insight into how additional sources could positively improve the CAR process. From work reviewed, PDNS was identified as being an overlooked and undermined source of information, promoting a drill down into related research. This further review of research concluded that the predominantly use case of PDNS is in malicious domain name analysis (Xuanzhen, Zulie and Yuanchao, 2020). However, a dearth of research is evident in citing the use case of the PDNS in identifying dangling DNS records, highlighting a gap in current research, and identifying an additional source of useful information to correlate with IWS data to improve upon asset discovery and risk assessment capabilities.

CHAPTER 3

3 METHODOLOGY

3.1 INTRODUCTION

In this chapter, the methodology will outline the design and implementation of an intuitive web-based non-intrusive asset discovery and known-vulnerability assessment tool. The design and overall implementation will be heavily inspired by related tools found in current literature. In line with the paper objectives, only free publicly available data sources will be considered.

Furthermore, this section will outline the testing methodologies and processes to be adopted during the experimentation phase of the tool, ensuring the original specification is met and the tool is working as intended.

3.2 DESIGN

This section will discuss the proposed approach, with an in-depth detailed breakdown of each fundamental component. The latest stable version of Python 3 was selected as the programming language of choice for backend development. Python was selected due to its exhaustive list of related dependencies and extensive documentation identified throughout the literature. Prior knowledge and experience working with this language was also beneficial.

As with any application development, the key inputs, processes, and outputs were first noted down to break the project up into manageable chunks. These can be seen in the high-level Input-Process-Output (IPO) diagram in Figure 17.

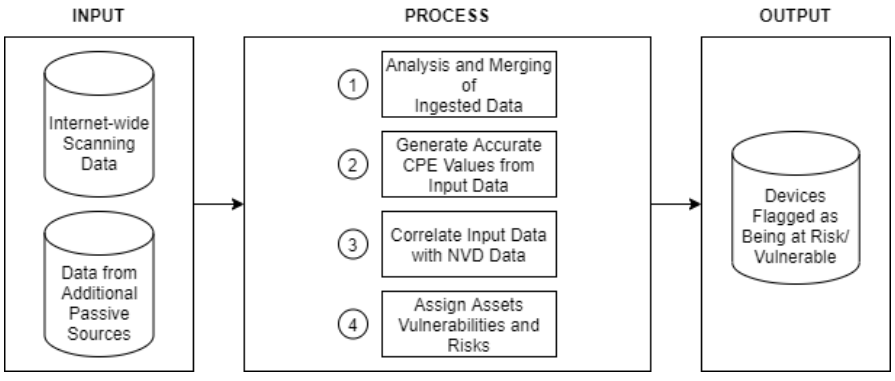


Figure 17: IPO Diagram for Tool

3.2.1 Inputs

The input stage consists of many inputs categorised into two categories, IWS platforms and additional passive data sources. The purpose of the first category is to ingest data from the Shodan, Censys, BinaryEdge and Onyphe platforms, each of which have been selected based on the in-depth review and analysis conducted within the literature review. The primary reasons attributing to the selection of each platform was down to the quality of available documentation and prior proven capabilities.

The concept of correlating multiple IWS sources is a key feature setting this tool apart from the related ShoVAT and Scout tools, which solely rely on one IWS data source. (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019). This feature is vital in addressing the issue of IWS probes being blocked and mitigating issues surrounding the unique limitations posed by individual platforms. Additionally, obtaining data from a multitude of sources ensures that the most up to date data available is used to construct a comprehensive picture of an asset, rendering another key issue on the timeliness of IWS results highlighted by Li *et al.* (2021) less of a concern for the tool being developed. It was decided at this stage that in the interest of saving development time, each platforms corresponding Python API wrapper library would be implemented into a handler class, if technically feasible, to avoid extensive, tedious API work. Each IWS platform will have its own handler class responsible for obtaining, storing, and normalising data.

From the literature reviewed, PDNS was identified as being an overlooked and underutilised OSINT resource within the security domain, which happens to correlate well with IWS data. Hence it was selected as the best option for an additional passive data source. Several free PDNS sources were selected as being adequate for this project based on the following criteria:

- Well-established and documented API
- Reliable uptime
- Freely available API features with minimal limitations

Details of each considered platform and how well they met the above criteria can be found within Table 7

Table 7: PDNS Source Comparison

			CRITERIA			
SOURCE	**API**	**Well Documented**	**Free Access**	**Requires Account**	**Uptime**	**Limits**
ThreatCrowd	Yes	Yes	Yes	No	100%	Limited
ThreatMiner	Yes	Yes	Yes	No	99.70%	Strict
Robtex	Yes	Yes	Yes	No	100%	Strict
Daloo	Yes*	No	Yes	No	100%	Lenient
FarSight	Yes	Yes	Yes	Yes	100%	Limited
DNSGrep	Yes	Yes	Yes	No	100%	Lenient

** The API is incomplete and not official supported/advertised*

From Table 7, it can be determined that the selected platforms have adequate free access, with each providing an interfaceable API. However, Daloo by Alexander Georgiev does not currently provide a complete API due to the nature of the platform being a hobby PDNS project. As such, no API documentation is available, but due to the simplistic nature of the site, web scraping for the desired information is a practical and easily achievable solution. Of the listed sources, only FarSight requires an

account to obtain access, which can be obtained for free. However, this free account is only intended to be a grace period; hence limitations are in place on the account. Robtex, ThreatCrowd and ThreatMiner are threat intelligence platforms that collate a host of information, including PDNS. Each of the platforms come with differing severities of API throttle limits that will need to be taken into consideration during implementation. The remaining platform, DNSGrep, pulls data provided by the trusted and reliable Rapid7 Project Sonar (Rapid7, 2021) datasets and efficiently returns the results via an API (Erb, 2019). Of all the data sources selected, this is by far the most conclusive and efficient. The uptime of all platforms was found to suffice, with uptime statistics being produced over a 30-day period of continuous automated monitoring from varying geographic locations (Uptrends, 2021).

Each data source will require a handler class to be created for obtaining, storing, and normalising the data for analysis at a later stage. A full breakdown of all input sources can be found in Figure 18. This includes all PDNS and IWS handlers.

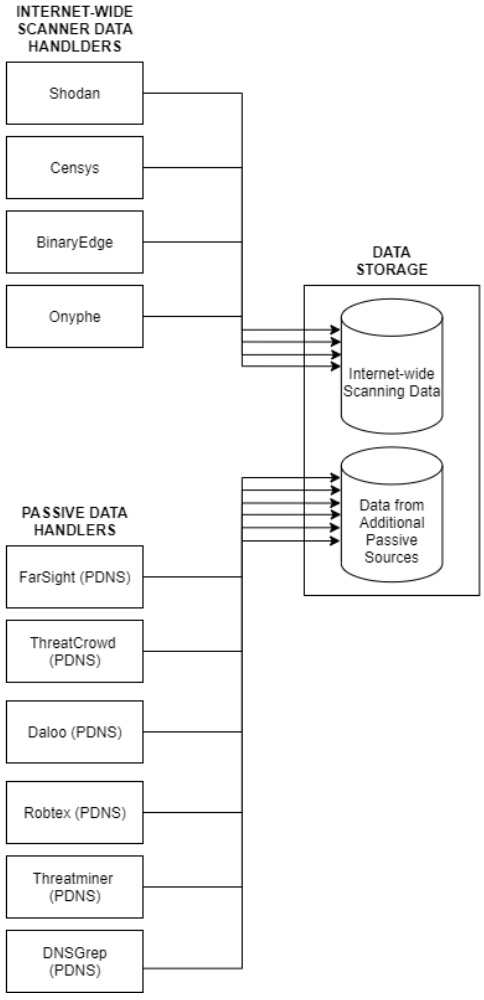


Figure 18: Input Source Breakdown

Data returned from each of the selected sources can contain a significant quantity of results with differing fields, formatting, and attributes. To handle this, each source handler will be implemented with a custom data normalising method specific to the structure of data being gathered, allowing for data to be formatted into one of the two standard formats in Figure 19. Whereby highlighted attributes are required,

and non-highlighted attributes are optional and dependant on available data. As the data obtained from sources differs vastly and has no set schema, MongoDB, a NoSQL non-relational database, was selected as the primary storage solution. Its non-structured NoSQL nature, good documentation, and reputation for high performance time-series big data handling made this an excellent choice.

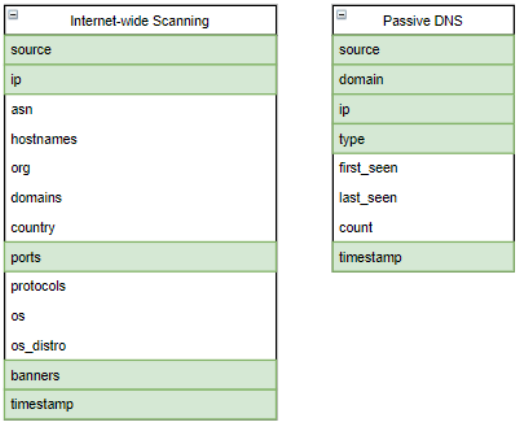


Figure 19: Field Standardisation for Data Source Handlers

End users will be able to query these ten input sources via a simple search bar accepting a valid IPV4 address or CIDR range. Additional complexity is to be added by allowing a degree of user search customisation, including the ability to adjust the maximum quantity of IWS records retrieved as well as allowing users to select which data sources they desire to utilise. Both adjustable parameters contribute to the increased usability of the tool and are vital in helping ensure end users do not exceed their API quotas.

3.2.2 Processes

The process stage can be split up into multiple sections. The first of which concerns the merging of normalised data ingested from all IWS sources, a process that will be solved using a simple but efficient in-house algorithm. This process aims to derive the latest, most relevant data from results returned by the users desired selection of data sources. The normalisation of data within the IWS handlers makes this stage in the process manageable, as the keys required to merge data are already standardised. However, this process is further complicated because IWS handler normalisation methods are not required to fulfil each key within every dataset obtained due to differing categories of data being returned by each selected platform. Although this design decision saves space and eliminates excess quantities of duplicate data, it vastly complicates the merging process. Secondly, the service banner analysis, CPE construction, and CVE assignment features of the tool will be designed, with considerations for flaws found within the NVD and a key focus on accuracy. Thirdly, the functions required to further non-actively enrich the PDNS data for it to be usable in the identification of dangling DNS records will be designed. The final element of the design phase will consider the methods required for adding additional risk assessment features to the tool.

3.2.2.1 Internet-wide Scanning Data Merging

The algorithm designed to tackle the problem of IWS data merging is required to be highly versatile as the only known factor is the required standardised IWS attributes, as highlighted in Figure 19. The total

number of data sources to be merged and the quantity of records provided by each is unknown and largely decided by the end user. To achieve the expected result, differing merging approaches will be required to be adopted for each attribute. Inspiration derived from the solution to the notorious 25 horses programming puzzle (Croak, 2019), aided with the design of this solution. The formed method follows an iterative approach, considering only two data sources at a time, allowing for a high level of versatility. The merging of most fields can be achieved by employing a combination of several logical comparison operations. However, as the integrity of service banner data is crucial to the success of this project, an additional more robust refinement solution was derived. During the final merging process, it is proposed that the service banner data is to be merged separately based on timestamp data. For example, if two platforms detect a service on port 22, the service banner with the oldest detection timestamp would be dropped from the merged data. A high-level description of the formed merging approaches to be implemented into the tool can be found in Table 8.

Table 8: IWS Attribute Merging Approaches

Field	Approach
source	No merge required.
ip	No merge required.
asn	Select the most reoccurring value. If multiple, select the initial source value.
hostnames	Combine values and drop duplicates.
org	Select the most reoccurring value. If multiple, select the initial source value.
domains	Combine values and drop duplicates.
country	Select the most reoccurring value. If multiple, select the initial source value.
ports	Combine values and drop duplicates.
protocols	Combine values and drop duplicates.
os	Select the most reoccurring value. If multiple select initial source value.
os_distro	Select the most reoccurring value. If multiple, select the initial source value.
banners	Combine values and drop duplicates based on the most recent timestamp data available.
timestamp	No merge required.

3.2.2.2 Service Banner Analysis

Upon successfully merging IWS data, the next stage is to design an effective service banner analysis algorithm to generate accurate CPEs based on the approaches adopted in the previously reviewed literature. As already discussed, CPEs associated with services can often be incorrect or depreciated, a factor that only two papers took into consideration when designing a CPE construction algorithm (Sanguino and Uetz, 2017; O'Hare, Macfarlane and Lo, 2019). The authors Sanguino and Uetz (2017) approach the problem of mitigating the adverse effects of NVD errors by implementing a third-party software information database to assist with generating accurate CPEs. Further coupling this with the Levenshtein distance algorithm to resolve the problem of misspelt CPEs enabled this approach to produce

a list of relevant CPEs. Using this list, the most suitable CPE was determined based on version information. O'Hare, Macfarlane and Lo (2019) further refined the previously discussed approach by considering other approaches and removing the reliance on a third-party software details database. In addition to ensuring a sole CPE was selected based on if the returned length of the Levenshtein algorithm edit distance was of a particular value. Instead of returning multiple possible CPE matches. Such an approach removed the reliance on accurate version information needing to be obtained, as is the case with the past discussed approach by Sanguino and Uetz (2017). This can be seen as a considerable advantage due to the general lack of version information obtainable from banners gathered by IWS platforms.

Both solutions by Sanguino and Uetz (2017) and O'Hare, Macfarlane and Lo (2019) consider NVD flaws and are similar in their approaches, although the second more refined approach is the more simplistic and efficient of the two. However, both failed to tackle depreciated CPE values, a problem that no approach found within literature addresses. To mitigate this problem, a further refined algorithm with the inclusion of a CPE exception list was designed primarily based on previous work by O'Hare, Macfarlane and Lo (2019). Additionally, this solution will utilise the IWS platforms capabilities for deriving individual attributes of service banners instead of attempting to determine this information locally, as is done by current tooling within literature. Theoretically, passing this task off to the IWS platforms should improve the overall reliability of the attribute separation process due to IWS scanning platforms having more information on the target and better capabilities to facilitate the operation. Furthermore, this approach removes an additional layer of complexity from the CPE creation process, increasing its accuracy and ability to detect CPEs of uncommon service banners. A high-level overview of the formed approach can be visualised in the flowchart found in Figure 20. The algorithm works by taking the initial CPE generated from service banner attributes and refining it against a selection of exception checks, before further validating it using a ratio in conjunction with the Levenshtein algorithm. Upon a successful match, the validated CPE is associated with CVEs in accordance with data from the NVD.

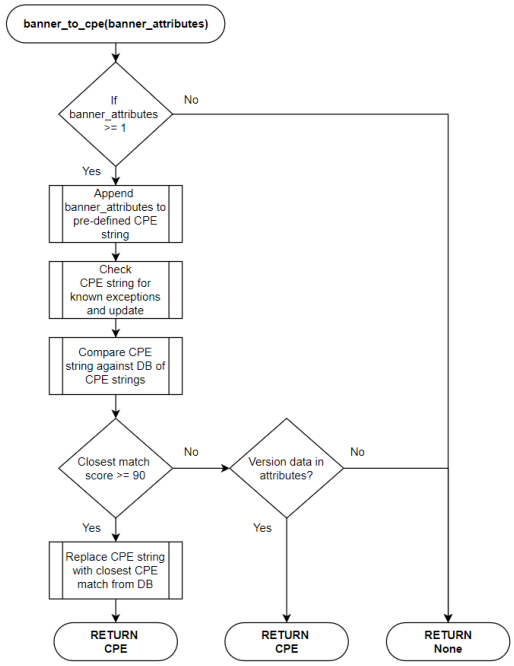


Figure 20: CPE Construction Algorithm

3.2.2.3 Passive DNS Enrichment

Unlike the previous CAR tools circulating within reviewed literature, this developed solution will implement PDNS as an additional passive data feed, enabling unique data pivoting possibilities that can be leveraged to further discover or risk assess assets. However, upon ingesting and normalising the PDNS data, further passive enrichment will be required to achieve the desired functionality. Specifically, the ASN of every IP address gathered from PDNS will need to be obtained to allow for dangling DNS record checks via cross-referencing with the original scan ASN. Team Cymru provide an excellent free service for mapping IPs to their corresponding ASN (Team-Cymru, 2021). The implementation of this feature will allow for potential dangling DNS records to be identified and for the further asset discovery of potential lost and forgotten assets. An illustration of this enrichment process can be seen in Figure 21.

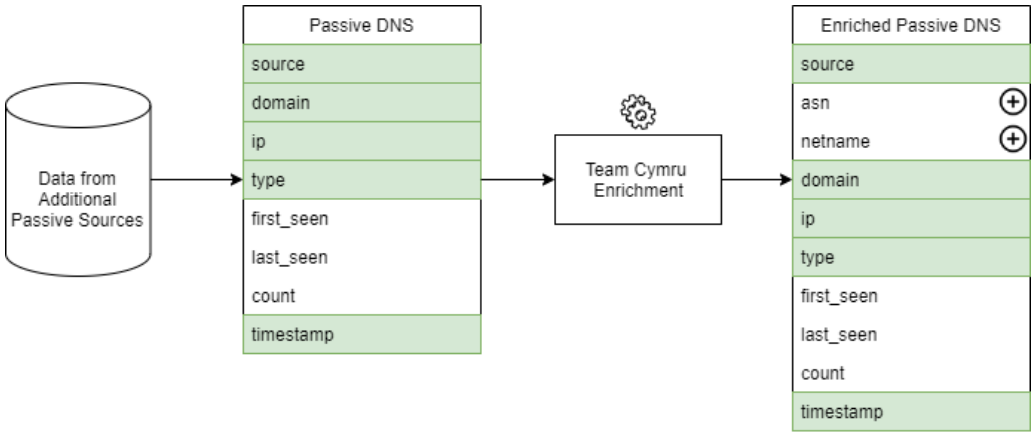


Figure 21: PDNS Enrichment Process

3.2.2.4 Default Vulnerabilities, rDDoS and Port Assessments

Further exploring avenues that IWS data can be utilised, it is proposed that components within the CVSS 3.1 vector specification could theoretically be used to identify CVEs exploitable with default configurations. Although, due to the known substandard quality of the NVD dataset and the high frequency of inaccurate data, accurately determining such a factor will not be easy. Thus far, no study has come up with a reliable approach to determine such a factor. Therefore, two approaches to achieving this are proposed. Only one of which will be implemented and evaluated for effectiveness during this study due to time constraints. The first proposed approach is to build upon and repurpose CVE text-mining research covered within the literature review (Guo, Xing and Li, 2020; Wåreus and Hell, 2020), with the goal of determining if associated summary data is enough to determine if a CVE is vulnerable by default. The second proposed approach is to take advantage of the aforementioned CVSS 3.1 vector specification, specifically investigating if the attack complexity and attack vector components are viable indicators for determining default config vulnerabilities. This secondary approach will likely be the solution integrated, due to project time constraints.

A further addition to the tool will be the capability to predict the maximum rDDoS throughput of assets, based on work produced by the authors Leverett and Kaplan (2017), who developed a methodology for calculating the worldwide capacity for rDDoS attacks. Based on this work, the maximum rDDoS potential of services can be determined based on port data obtained from IWS platforms in correlation with the

maximum Bandwidth Amplification Factor (BAF) of services. Such figures can be observed in Figure 22. However, IWS port data will be the only determining factor for service identification for this feature, which has drawbacks due to three bold assumptions. The first being that the services identified are running on their default ports. Secondly, that the IWS platform of choice is frequently scanning the ports. Thirdly, that service version factors or mitigations in place on services known to be affected are not taken into consideration. Thus, rDDoS is assumed feasible by default which is not always the case. Due to these reasons, this solution should only be used as an estimated gauge for the theoretical maximum predicted rDDoS BAF throughput.

Protocol	Bandwidth Amplification Factor	Vulnerable Command
DNS	28 to 54	see: TA13-088A [4]
NTP	556.9	see: TA14-013A [5]
SNMPv2	6.3	GetBulk request
NetBIOS	3.8	Name resolution
SSDP	30.8	SEARCH request
CharGEN	358.8	Character generation request
QOTD	140.3	Quote request
BitTorrent	3.8	File search
Kad	16.3	Peer list exchange
Quake Network Protocol	63.9	Server info exchange
Steam Protocol	5.5	Server info exchange
Multicast DNS (mDNS)	2 to 10	Unicast query
RIPv1	131.24	Malformed request
Portmap (RPCbind)	7 to 28	Malformed request
LDAP	46 to 55	Malformed request [6]
CLDAP [7]	56 to 70	—
TFTP [23]	60	—
Memcached [25]	10,000 to 51,000	—
WS-Discovery	10 to 500	—

Figure 22: Bandwidth Application Factors (CISA, 2019)

The final risk assessment feature to be implemented is a simplistic but effective user-definable high-risk port list. Port information gathered by IWS platforms will be compared against this customisable user predetermined list of high-risk ports and flagged appropriately. This feature adds an additional layer of risk assessment. It is a feature that can be especially effective in scenarios where significant delays in official CVE assignment is encountered, an issue that is prevalent within the NVD (Rodriguez *et al.,* 2017). Manually allowing users to flag potential risks based on port exposure.

3.2.3 Output

All data provided by the backend will be outputted in a meaningful, visually appealing, easy-to-understand format via a front-end web interface, mainly in the form of interactive data tables with search and sorting capabilities. The popular Python-based micro web framework, Flask (2021), will be utilised to handle the web logic aspect of the project in a timely and secure manner. Flask was selected over competing frameworks due to its easy learning curve and more lightweight nature compared to other popular frameworks like Django (2021).

The web-based front-end pages will be written primarily from scratch utilising a combination of HTML, CSS, and JavaScript components in conjunction with bootstrap and several free assets from Font Awesome. The outputted structure of data is critically important in ensuring identified risks are instantly identifiable to end users. Additionally, a smooth and efficient user experience adds to the overall

useability of the tool. Hence the final application will have as few navigable pages as possible to simplify useability. These pages will come under three primary categories: dashboards, projects, and drilldowns.

The primary display output method for data will be via data tables due to their highly flexible and reusable nature. Allowing for the component to be reused throughout the project regardless of the data needing to be displayed, saving development time. As can be derived from the input and process sections above, a total of two normalised sources of data needed to be displayed, both of which have a minimum of one common field. As such, by utilising the complete flexibility of data tables, both normalised sources of both IWS and PDNS data can be merged into a single data table if desired. The scale and size of data is another key factor to consider during the design phase, especially concerning performance. Traditional means of passing, handling and rendering data client-side, although relatively efficient with small datasets, is not a practical solution when dealing with large datasets as is the case in this scenario. The datasets will need to be rendered by a set of paginated data tables with all data processing operations handed off to the server to overcome this. To achieve this smoothly, AJAX will be used for asynchronous communication between server-side functions handling the required data and the front-end interface. Additionally, automated statistical overviews, including graphs, node maps, and numerical values, will be included by leveraging the power of JavaScript.

UI wireframe diagrams were produced to approach the web design aspect of this project. This aided with envisioning the main pages, ultimately helping to ensure the overall usability and accessibility of front-end components. Regarding the web design approach adopted, the decision between a minimalism or maximalism design was drawn based on a study by Mejtoft, Carlsson and Söderström (2019) that set out to find if millennials would favour the minimalism approach over maximalism. The study found participants to be more enthusiastic about the maximalism approach, however, both approaches were well received, and it was concluded that the approach used is inevitably down to the message the site is trying to convey. Due to the proposed applications use case being in security, likely in a professional setting, the decision was made to adopt the clean minimalism approach to UI design. Figure 23 contains an initial conceptual design for the web application's home page based on these findings. Furthermore, design inspiration was sought from well-established modern billing and administrative portals (Pinterest, 2021).

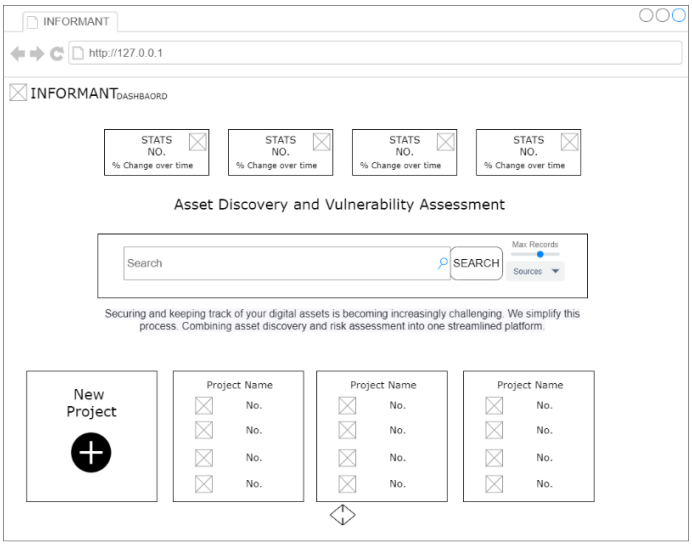


Figure 23: Dashboard Wireframe UI Design

To improve upon similar command-line-driven tools, Scout and ShoVAT, which have been drawn upon in previously reviewed literature. This solution aims to stand out by offering end users a host of features aiming at analysing and visualising data via a web interface. During this phase, a variety of relevant but simple names were considered for the proposed tool. Resulting in the proposed name of Informant being selected as the best fit for the tool due to relevant connotations with the tools non-intrusive reconnaissance nature.

3.2.4 Architecture

In order to serve and deploy the web application, a comprehensive backend web architecture needs to be designed. The proposed core architecture primarily consists of a MongoDB instance for data storage, a Redis, in-memory message broker instance to handle task queues, and Nginx, a complete web server solution. The proposed full-stack architecture can be seen in Figure 24. To ensure a clean and secure development life cycle is followed, the GitHub (2021) platform will be utilised to ensure consistent version control throughout.

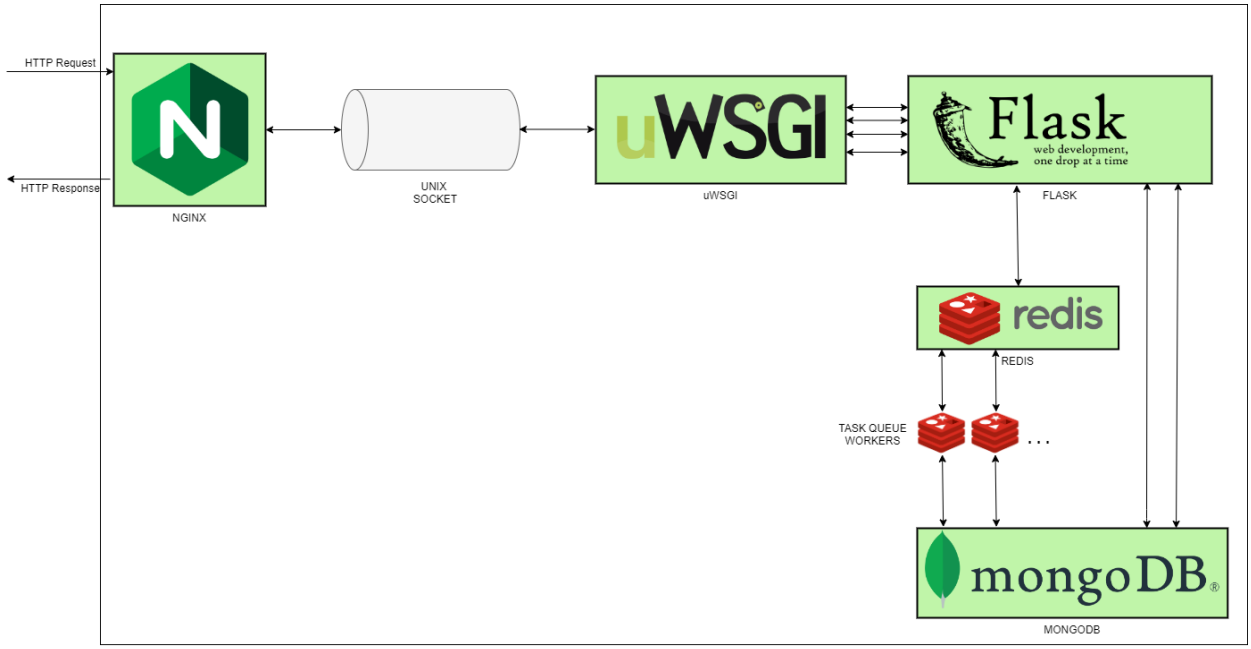


Figure 24: Full Stack Architecture

3.3 IMPLEMENTATION

This section will draw upon crucial decisions made during the implementation of the previously discussed tool design. The project relies on several third-party platforms to provide reliable data streams. Therefore, as discussed in the design phase, where applicable, each corresponding pre-existing Python wrapper codebase will be integrated to help develop this tool in a timely manner. A modular, agile approach to development was adopted, focusing on breaking down the project into manageable achievable tasks, which can then be tested and combined progressively (UKEssays, 2019). The design section broke down and lay the groundwork for the development process, splitting it into several manageable input, process, and output tasks.

Continuous unit testing throughout the implementation of each task during development will ensure that the eventual integration process of all code sections will run smoothly and lessen the overall time spent debugging. Additional integration testing will also be conducted to ensure newly developed modules function as expected upon implementation into the core project, ensuring no adverse effects on the pre-existing codebase. As mentioned in the design phase, GitHub will be employed to manage version control during the implementation stage efficiently. Regular commits will be pushed to GitHub to ensure progress is documented and backed up, adding a layer of data redundancy, ensuring that in the event of local data storage failure, minimum progress would be lost.

To ensure a high level and consistent quality of code, the principles underlined in The Hitchhiker's Guide to Python Best Practices for Development (Reitz and Schlusser, n.d.) will be adhered to. This will help ensure satisfactory code readability, understandability, and adaptability. Three factors that are often associated with good code and factors that predominantly affect the future maintainability of code. The cohesion and coupling of code are also essential factors to consider concerning the quality of code. The cohesion of code relates to how close modules relate to each other, whereas the coupling of code refers to the strength of module relations. Therefore, the developed tool will aim to achieve a weak coupling and a close cohesion, meaning that components are highly independent of each other and are capable of solving a particular part of the overall problem being tackled by the tool.

3.3.1 Internet-wide Scanning Handlers

Following a modular design from the onset, each of the four IWS platforms was encapsulated within their own handler class in accordance with the design phase. The official comprehensive, supported Python API wrapper packages for Shodan, Censys and BinaryEdge were used to efficiently query and fetch data from each platform. The use of these libraries significantly advanced the development pace. Unfortunately, although the final Onyphe platform has a Python API wrapper available, it was found to be exceedingly barebones and not well maintained. As such, the decision to write an in-house API wrapper for Onyphe was made to allow for additional flexibility and easier customisation.

The structure of each IWS handler was maintained by constructing a boiler template with the bare minimum required methods and attributes for each. These methods and attributes can be seen in Figure 25. Using this template, each handler was developed whilst continuously referring to their associated API and library documentation where applicable. As shown in the boiler template, each IWS handler accepts two parameters for the search function, query and max records. Both inputs are customisable by the end user, with the query parameter representing the desired target, accepting IPv4 addresses and CIDR ranges as valid inputs. Whilst the max records parameter accepts whole numerical values and determines the maximum quantity of records allowed to be retrieved from any given IWS platform, a feature that will help ensure credit limits are not accidentally exceeded. Crucial to the success of the experimentation phase, in the allocated time frame.

```python
class IWSHandler:
    """
    Main class to retrieve information from ___ API.
    """
    def __init__(self, api_id: str = StandardValues.___, api_secret: str = StandardValues.___):
        self.results: list = []
        self.resultTotal: int = 0

    def search(self, query: str, max_records: int = StandardValues.____):
        """
        Function used to search hosts
        """

    def formatted_results(self):
        """
        Function to return formatted results
        """

    def raw_results(self):
        """
        Function to return raw results
        """
        return self.results


    def total_results(self):
        """
        Function to return total sum of results
        """
        return self.resultTotal
```

Figure 25: IWS Handler Boiler Template

Although each IWS platform handler implementation followed the boiler template, each differed slightly due to varying input parameters and formatting differences associated with their APIs. Each of the handlers was given access to a standard values class where all configurable non-sensitive options are held. The standard values class can directly communicate with the central MongoDB database, responsible for storing the platform API keys, enabling settings to be changed on the fly. This class can be found in Appendix M. Unlike the comparable tooling, which retrieves the entirety of the service banner in a pre-congealed string format, the developed IWS handlers ingest pre-separated banner attributes. This gives Informant a distinct advantage over Scout and ShoVAT by avoiding the need for additional banner analysis to identify individual attributes from service banner strings (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019). The following few paragraphs will highlight how the unique challenges attributed to utilising freely available accounts on each platform were overcome during the implementation stage.

For Censys, two API endpoints were considered to retrieve the desired results for the Informant tool. Firstly, the search API which returns a paginated list of results with a finite amount of information based on a given query. Secondly, the view API that fetches a full data dump of information relating to an individual entity. Queries on both these API endpoints require one credit to be deducted from the Censys monetised credit system, meaning that a significant quantity of credit would be required to use both API endpoints in tandem effectively. This would be required to obtain all the desired information on each device detected by Censys fully. For example, querying '192.168.0.0/27' via the search API using a free account allows for the retrieval of 1,000 results per query, costing a total of one credit. Therefore, all 32 potential IPs within the CIDR range can be retrieved in one request costing only one credit. However, to obtain the complete detailed information surrounding each of the IPs observed from the original search query, a further 32 queries and thus credits are required to utilise the view API endpoint. As such, this approach was deemed impractical and far too credit intensive for this project. In this case, the alternative, more adequate solution was solely to use the search API endpoint and related data. However, the data available from this API call is not as substantial as the view API, with a limited selection of twenty related

attributes per IP address allowed (Censys, 2021e). Resulting in having to limit information gathering by Censys to the following attributes found in Figure 26.

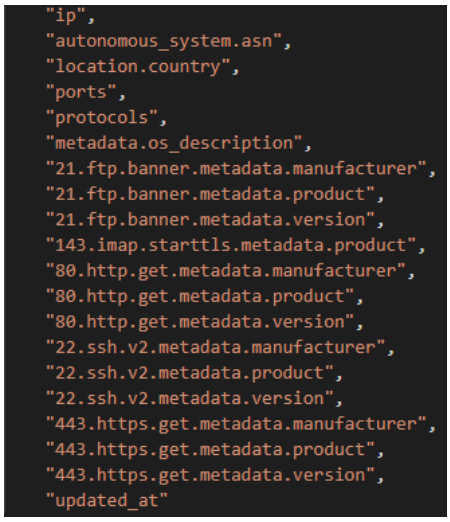


Figure 26: Censys Search API Attributes

Although this approach limits the tools ability to retrieve the full quantity of data on offer by Censys, it is a necessary sacrifice to ensure minimal disruption during the experimentation stage due to credit limitations. If this project were to have financial support, such a limitation would have been avoidable, leaving room for future development. Additionally, as discovered in the literature review, obtaining a research account on the platform does not give additional credits; hence it was not deemed necessary to do so.

Shodan, similarly to Censys, also required two API calls to obtain the desired results. However, in the case of Shodan, due to a more lenient credit system in place and generous academic account offerings, all required data held on each device can be obtained without significant credit usage. This can be achieved by leveraging both the '/shodan/host/search' API call to identify all responding IP addresses and the '/shodan/host/{ip}' API call to gather a complete detailed report on each responding IP iteratively. This combination of API calls allows for queries to be conducted on IP addresses and CIDR ranges whilst utilising minimal credits. As this approach is iterative, it can result in a significant number of '/shodan/host/{ip}' API calls, which could potentially lead to rate limiting. A hardcoded 1-second delay between requests was implemented as recommended within the official API documentation to prevent this.

The BinaryEdge API requires a different approach than previously implemented to extract the information needed in a credit efficient manner. Merely one API request is required to achieve the objective of scanning CIDR ranges and individual IP addresses for information. This can be achieved by utilising the '/v2/query/search' API endpoint, which returns a list of recent events based on query parameters. However, only twenty events are returned per page, which costs a total of one credit. Multiple events can correlate to the same IP address as BinaryEdge classifies services, ports and other related device data as individual events. Hence, if specific IPs have a plethora of data, several events per IP are expected, resulting in an incremental increase in credit usage. Therefore, to best utilise the credit system and implement the max record feature into the platform handler, the required number of pages must be calculated based on the max record value. This was accomplished by creating an iterative page-based query approach in conjunction with the BinaryEdge Python library. Although the approach is slightly

cumbersome, it works efficiently, superseding the alternative credit intense method of collecting all IPs and individually querying each one. The algorithm deployed to achieve this pagination max record round up to the closest page increment of twenty can be seen in Figure 27. To further optimise the credit intensity of this handler, the 'type:service-simple' filter is passed in with each query to ensure only the service identification module results are considered.

```
# Check if max_records is not an increment of 20
if (pages != StandardValues.BINARYEDGE_DEFAULT_RESULTS_PAGE):
    pages = (math.ceil(max_records/20))

# Converted to max_records e.g. 20 = 1 page, 8 = 1 page, 122 = 7 pages (round up)
page_index = 0
for x in range(pages):
    tmp_results: list = []
    page_index += 1
    try:
        if (page_index <=0):
            break
        else:
            tmp_records = self.api.host_search(query, page=page_index)
            if (tmp_records['total'] < max_records) :
                max_records == tmp_records['total']
            tmp_results = list(islice(tmp_records['events'], max_records))
            for y in range(len(tmp_results)):
                self.results.append(tmp_results[y])
            max_records -= 20
            page_index -= 1

    except (BinaryEdgeException, BinaryEdgeNotFound) as apiError:
        print(f"[BE] BinaryEdge API error: {apiError}")
```

Figure 27: BinaryEdge Pagination Handling

Onyphe the final IWS handler to be implemented was the only platform in which a third-party API wrapper was not utilised to speed up the development time. Instead, a custom wrapper following the API documentation was built based on the requests Python library, simplifying the process of working with HTTP requests. Unlike the other platforms implemented, the free account on Onyphe lacks the capability of allowing CIDR scans, a key feature of Informant. As a result, this platform will be exclusively used for data enrichment on previously detected IPs by the alternative IWS platforms. Therefore, requiring the Onyphe handler to be run in conjunction with at least one other scanner as a list of IPs produced from previous scans will need to be obtained and duplicates removed. The compiled list of unique IPs will be iterated through retrieving the desired data from the '/summary/ip' API endpoint. A credit is deduced for each IP checked, hence caution with this IWS must be taken when scanning large subnets as the free account is limited to 250 credits per month.

Corresponding unit tests were constructed for each handler class to ensure their correct operation and that data being retrieved was as expected. An example of the output produced from a CIDR query limited to a maximum of two records can be seen in the output produced by the Censys and Shodan handler unit tests in Figure 28 and Figure 29, respectively.

```
[CENSYS] Censys successfully authenticated.
{'location.country': 'United Kingdom', 'ip': [redacted], '22.ssh.v2.metadata.product': 'OpenSSH', 'updated_at': '2021-03-17T06:40:43+00:00', 'autonomous_system.asn': 42675,
'22.ssh.v2.metadata.version': '7.4', 'ports': [22], 'protocols': ['22/ssh']}
{'location.country': 'United Kingdom', 'ip': [redacted], '22.ssh.v2.metadata.product': 'OpenSSH', 'updated_at': '2021-03-17T12:15:03+00:00', 'autonomous_system.asn': 42675,
'22.ssh.v2.metadata.version': '7.4', 'ports': [22], 'protocols': ['22/ssh']}
```

Figure 28: Censys Handler Data Retrieval Output

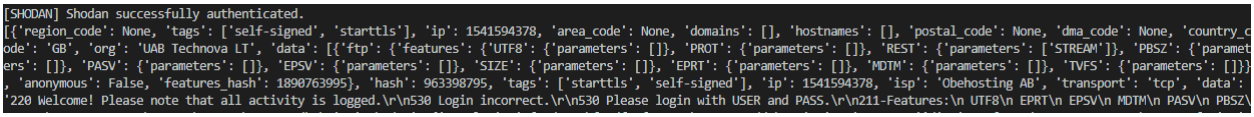


Figure 29: Shodan Handler Data Retrieval Output

Now that the retrieval process for each platform was complete, it was time to implement data normalisation algorithms in line with the planned design. As shown in Figure 26 and Figure 27, the data gathered is chaotic and, in the case of Shodan, rather excessive for the needs of this project. For this reason, an efficient but simplistic normalisation algorithm is needed to bring to light the required data retrieved from each scan. To develop this, the API JSON data schema documentation of each of the four IWS platforms were reviewed to identify the fields of interest as denoted in the standardised data structure design shown in Figure 19. Once identified, this data could be extracted and processed into formatted dictionaries ready for storage. An example of the formatted output produced by the previously shown raw Censys and Shodan results from this process can be seen in Figure 30 and Figure 31, respectively. Comparing the pre-processed and processed results, the data can be seen to be clear and concise, with all redundant data removed.

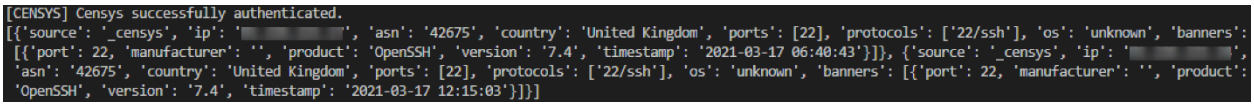


Figure 30: Censys Handler Formatted Output

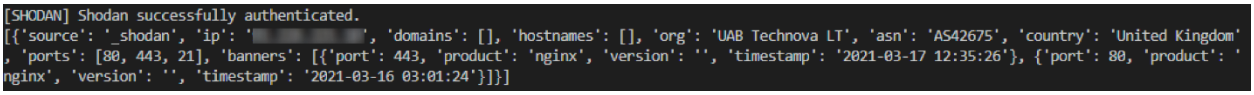


Figure 31: Shodan Handler Formatted Output

As stated in the design phase, not all attributes can easily be obtained from each of the four IWS platforms. The reasoning behind this varies from strict account limitations to the lack of available data and the increase in the number of credits required to obtain extra data. Additionally, due to the high level of complexity attributed to filtering out specific data elements, some aspects of retrieved data from platforms were ignored by the data processing approaches implemented if deemed nonessential to the tool's core functionality. This helped save copious amounts of development time. In Figure 32, a clear illustration highlighting the attributes retrieved from each platform can be observed. As previously stated, the exclusion of a given attribute is not necessarily down to a lack of information available by the platform. Such an exclusion is more than likely related to implementation time constraints or the requirement for further credit utilisation.

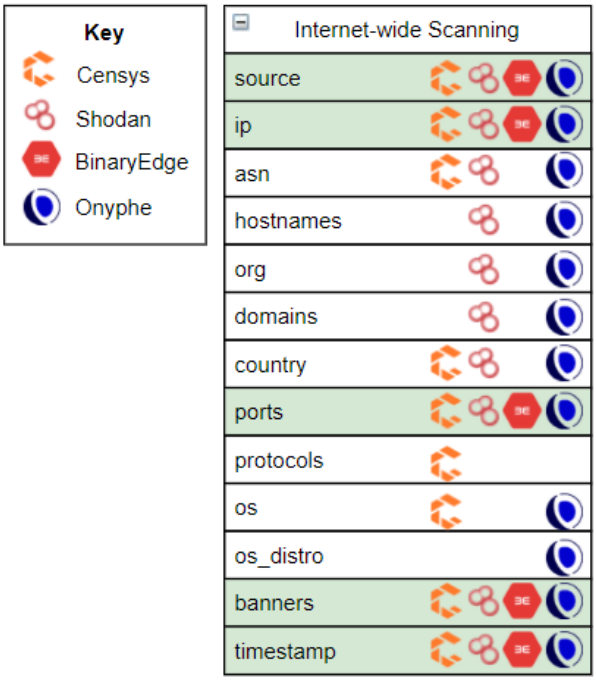


Figure 32: IWS Platform Attributes

Per the boiler template, the total results method was calculated by working out the sum of unique IP addresses retrieved from each IWS platform. Although platform-dependant, several other optional helper methods were implemented if deemed practical in a reasonable time frame. Such methods include account status checks to allow for periodic integrated remaining credit checks.

During rigorous unit testing of each handler, a mock test server was used as the target to help review and identify inconsistencies in results across platforms. Several programmatic data interpretation errors were identified and resolved during this process, and a meaningful outcome was alluded to. The integrity of attribute data received was found only to be as good as the methods employed by each IWS platform gathering the data in the first place. These methods are not always publicly disclosed and certainly do not come without flaws, with methods and techniques employed by platforms varying substantially. Investigating this issue in detail is out of this project's scope, as a whole separate paper could be written exploring the integrity and accuracy of methods deployed by IWS platforms. Thus, the need for further research in this area. In particular, the country geolocation data attribute is a prime example of when data integrity can be brought into question, as highlighted during unit testing. Such IP geolocation data is often faked by VPN and self-claimed bulletproof hosting providers. From the literature reviewed, fraudulent geolocation data is not a factor considered by any publicly accessible IWS platform, even though such data is widely faked.

All IWS platforms mentioned in this study rely solely on information obtained from regional internet registries to obtain ‘accurate’ country data. Such data is user-controlled and can be easily spoofed. Gathering information in such a way is straightforward but unreliable. However, it can be assumed that IWS platforms do not verify the information due to the lack of importance surrounding geolocation data compared to the other data gathered. Additionally, using current methods significant hardware resources are required to determine if location data is spoofed at scale accurately - a further deterrent for conducting such checks at scale.

Fortunately, the specific issue of detecting fraudulent geolocations has been addressed multiple times within literature, most recently by Williams (2020), who developed a tool capable of accurately detecting fraudulent IP addresses at scale. This was achieved by actively pinging and analysing latency response times from clusters of measurement servers located around the globe to triangulate an accurate estimated geolocation. Using this tool within a small segment of IPv4 space, 62,000 hosts were detected as having incorrectly identifiable geolocation data. This research not only proposes and implements a practical solution to the problem but also highlights how prevalent falsified geolocation data is.

As fraudulent geolocation data appears to be relatively commonplace - with it making an appearance during unit testing - a simplistic active based approach was implemented into Informant as an extra optional feature to verify the integrity of geolocation data. The developed approach was based on a simplified version of constraint-based geolocation (Gueye *et al.,* 2006). In this simplified approach, the client's physical location is used to determine the theoretical fastest time for a ping request to be returned based on the shortest distance between the host and the claimed IP origin. Using this distance, the theoretical fastest time for a ping request to be returned to the client by the responding IP is calculated, taking into consideration the maximum speed of light in glass – a key element within fibre cables widely used to transfer data. Using the formula in Equation 1, it can be determined if the ping request was returned faster than physically possible, allowing for a conclusion to be made on whether or not the IP geolocation data is fraudulent or not. Although this solution is not comprehensive, it can be used as an initial indicator for determining fraudulent geolocations. The average ping between Informant and the target IP is taken from three ping requests executed in quick confession and distance data is pulled from distance24 API (Distance24, 2021).

A drawback to this approach is that no considerations are made for other factors that can influence latency timings. However, it can confidently be determined that a response cannot return faster than the rules of Physics allow. Therefore, the false positive rate will be low whilst the false negative rate will be higher. The more accurate solution implemented by Williams (2020) would vastly lower the margin of false negatives; however, it is deemed as excessive and far too resource-intensive for this project.

Equation 1: Informant's Geolocation Validation Formula

$$L_m = \left(\frac{D_m}{c}\right) x\ 1000$$

Where the L_m denotes the theoretical fastest possible ping response in milliseconds, c denotes the estimated average speed of light in a fibre optic cable and D_m denotes the vector distance between the base location and claimed target destination.

Unit testing of this feature found it to successfully flag the fraudulent geolocation of the test server IP address being used. The IP address in question is registered under the Asia Pacific Network Information Centre and displays a falsified geolocation of Myanmar. However, its actual known physical location is within datacentre in the Netherlands. The results from this test and the code can be found in Appendix B.

3.3.2 Passive DNS Handlers

Continuing with the modular design philosophy, handlers for each of the six PDNS sources were developed. In order to efficiently gather data from the selected PDNS sources, a wrapper for the API of ThreatCrowd, ThreatMiner, Robtex, FarSight and DNSGrep was developed utilising the same Python

requests library used to interface with the Onyphe API. The remaining selected PDNS source, Daloo, was found to have no publicly documented API available during the design phase. As such, an efficient page scraping method was adopted to obtain the necessary data from this source.

A similar approach to the implementation of IWS handlers was taken during this stage, with the process being similar to the Onyphe platform handler integration due to a lack of efficient premade API wrappers. Although more tedious and time-consuming, developing in-house wrappers has the advantage of lessening the number of tool dependencies and allowing for less redundant code, as the developed wrappers only consider the required API endpoints. For this task, the IWS handler boiler template in Figure 25 was recycled with minimal name convention adjustments to make it suitable for PDNS data. Figure 33 shows the boiler template used.

```
class PDNSHandler:
    """
    Main class to retrieve information from ____ API.
    """
    def __init__(self):
        """
        Initialize ____ Class
        """


        self.results: list = []
        self.rdns: list = []
        self.resultTotal: int = 0


    def search(self, ips: list):
        """
        Function used to search hosts using ____
        API Docs @ ____
        """
        for ip in ips:
            . . .

    def searchCIDR(self, scan_range):
        """
        Function used to search hosts using ____
        API Docs @ ____
        """


    def outResults(self):
        """
        Return raw data
        """
        return self.results
```

Figure 33: PDNS Handler Boiler Template

The handlers for the five sources with available APIs were quickly developed and unit tested by frequently referring to each available API documentation and employing the template in Figure 33 above. These handlers included the implementation of data normalisation methods. An example of output from FarSight can be seen in Figure 34.

```
[FARSIGHT] Farsight successfully authenticated.
[{'source': '_farsight', 'count': 975, 'last_seen': '2021-03-20T07:45:15Z', 'first_seen': '2020-12-02T11:18:54Z', 'ip': '[redacted]
', 'type': 'pdns', 'domain': 'abertay.co'}, {'source': '_farsight', 'count': 381, 'last_seen': '2021-03-19T14:51:26Z', 'first_seen
': '2020-12-02T21:01:25Z', 'ip': '[redacted]', 'type': 'pdns', 'domain': 'api.abertay.ac.uk'}, {'source': '_farsight', 'count': 7
, 'last_seen': '2021-03-09T06:19:43Z', 'first_seen': '2020-12-17T10:33:29Z', 'ip': '[redacted]', 'type': 'pdns', 'domain': 'beta.
abertay.ac.uk'}, {'source': '_farsight', 'count': 10, 'last_seen': '2021-03-02T23:15:37Z', 'first_seen': '2020-12-08T00:48:58Z', 'ip
': '[redacted]', 'type': 'pdns', 'domain': 'sportlive.abertay.ac.uk'}, {'source': '_farsight', 'count': 10, 'last_seen': '2021-03
-02T23:15:02Z', 'first_seen': '2020-12-17T10:32:07Z', 'ip': '[redacted]', 'type': 'pdns', 'domain': 'abertaybeta.abertay.ac.uk'},
 {'source': '_farsight', 'count': 8, 'last_seen': '2021-03-02T23:12:20Z', 'first_seen': '2021-02-03T16:18:37Z', 'ip': '[redacted]
', 'type': 'pdns', 'domain': 'www.sportlive.abertay.ac.uk'}, {'source': '_farsight', 'count': 3, 'last_seen': '2021-01-19T04:40:38Z'
, 'first_seen': '2020-12-25T09:20:38Z', 'ip': '[redacted]', 'type': 'pdns', 'domain': 'vmwcmw01.uad.ac.uk'}]
```

Figure 34: FarSight Handler Output

In contrast to the IWS handlers, which are required to process several user inputs, none are considered by the PDNS handlers. All sources and thus corresponding handlers can facilitate reverse DNS lookups via IP addresses, with FarSight being the only selected source capable of CIDR reverse DNS lookups. The ThreatCrowd, ThreatMiner, Robtex, Daloo, FarSight and DNSGrep handlers only accept IPv4 addresses as a valid search parameter; hence each handler accepts a list of IPv4 addresses – which is obtained from IWS platforms. The implemented FarSight handler is the only PDNS source to require an API key and can accept CIDR ranges if provided within the project parameters. Alternatively, a list of IPs can be provided instead, and an iterative based approach can be used. However, the iterative process is credit intensive and ill-advised. As such, solely the developed FarSight CIDR range data retrieval feature will be used during experimentation to avoid hitting credit-related limitations. As an additional safety measure, in the event a CIDR range is not provided to FarSight, the function in Figure 35 will be used to estimate one, further mitigating the risk of running out of credits.

```
def ip_to_cidr(ips):
    ips.sort()
    startip = ipaddress.IPv4Address(ips[0])
    endip = ipaddress.IPv4Address(ips[-1])

    try:
        return [ipaddr for ipaddr in ipaddress.summarize_address_range(startip, endip)]
    except Exception as e:
        print("[INFORMANT] Failed to generated CIDR from IP range")
        return ips
```

Figure 35: IP List to CIDR

DNSGrep was the only platform selected for both reverse and forward DNS data gathering due to ease of implementation and time constraints. Such forward DNS data is crucial in assisting in the detection of potentially dangling DNS records. The implemented handler for this source considers both IP and top-level domain data as required. Such data is derived from previously retrieved IWS and PDNS data contained within MongoDB. A breakdown of each platform and corresponding data retrieved and considered by Informant can be found in Figure 36. As with the IWS handlers, the exclusion of a given attribute by a source is not necessarily down to the lack of information available and likely related to implementation time constraints or the requirement for further credit utilisation.

Key	
	ThreatMiner
	ThreatCrowd
	FarSight
	Daloo
	Robtex
	DNSGrep

Passive DNS Handlers								
	source	domain	ip	type*	first_seen	last_seen	count	timestamp
Reverse DNS	✓	✓	✓	✓	✓	✓	✓	✓
Forward DNS	✓	✓	✓	✓				✓

Figure 36: Passive DNS Handler Attributes

* Represents a static hardcoded value of 'pdns' which is used by the backend logic. This does not represent the DNS record type.

Depending on the number of IP addresses detected by IWS platforms, dozens of lookups will be conducted on each source, potentially causing the sources to throttle or lock out the client connection. To help avoid

this, configurable time-related mitigations have been put in place within the ThreatMiner, Robtex, ThreatCrowd, Daloo and FarSight handlers. All sources were identified as having connection related issues during unit testing, with the Robtex API being the most troublesome. Continuous unit testing was performed to fine-tune the time-based mitigations to resolve these connection issues. As already mentioned, the lack of API provided by Daloo called for a page scraping methodology to be adopted. However, thankfully the relatively simple nature of the site made such a task manageable using a series of regular expressions. The Daloo data retrieval works by utilising the request library to fetch the required pages containing the results. The desired attributes from these pages can then be extracted and refined locally via regular expressions. Avoiding the need for additional, often complex third-party web scraping libraries.

3.3.3 Data Storage

After developing the data handlers, capable of efficiently retrieving and normalising data from several selected sources, it was crucial to ensure that the obtained data was stored correctly to avoid data duplication, loss, and performance-related issues. Losing significant quantities of data due to data storage issues would have a significant adverse effect as credits used to gather such data are highly limited, leaving minimal room for error during this process. Therefore, to achieve efficient data storage between Python and the selected data storage solution, MongoDB, the well-established PyMongo library, was utilised to allow the core logical components to interface with the MongoDB API (Dirolf and Bernie, 2021). Additionally, for efficient MongoDB querying on the web application side of things via Flask, the Flask-PyMongo library was employed to bridge the communication between Flask and the data storage solution with several helper constructors.

For this project, all data gathered and generated by Informant is stored within one core database. The only other dependant database is a local copy of the NVD that is not necessarily required. However, it is fundamental in enabling primary known-vulnerability asset risk assessment functionality. As the data storage solution is non-relational, the data structure within the core database takes advantage of the collection's concept to segregate and organise individual scanning projects, allowing for efficient data management within each project. Each collection is made up of time-series data in the form of countless documents ingested from data retrieved by IWS and PDNS handlers, with care taken to ensure each document contains the least amount of duplicated information feasible. Such optimisations ensure that no document ever exceeds the maximum document limit of 16 megabytes (MongoDB, 2021). Each collection contains a master document containing meta-data relating to each stored project on the system and follows the structure in Figure 37.

```
_id: ObjectId("604641d64b51eb7a35040b1c")
origin_scan_range: "127.0.0.1"
last_scan_range: "127.0.0.1"
project_name: "Project 1"
› origin_params: Object
⌄ last_params: Object
    Shodan: true
    Censys: true
    BinaryEdge: true
    Onyphe: true
    Threatcrowd: false
    Threatminer: false
    Robtex: false
    Daloo: false
    Farsight: false
    DNSgrep: false
  max_records: 1
  created_at: 2021-03-08T15:25:00.000+00:00
  last_run: 2021-03-15T18:24:00.000+00:00
⌄ last_stats: Object
    totalAssets: 1
    secureAssets: 0
    risks: 1
    defaultConfig: 0
```


Figure 37: Project Meta-data Document

This approach to data storage allows for large quantities of small-sized documents to be stored within collections, allowing for high quantities of data to be stored whilst retaining the level of performance expected. Such an approach is more than capable of handling millions of documents within a single collection running on a singular MongoDB instance, assuming adequate hardware is in place. Thus, large quantities of PDNS data would not negatively affect backend database performance. Performance and scalability can be further improved by utilising sharding to distribute the data over multiple MongoDB instances. This would exponentially increase the storage capability of the tool but is deemed unnecessary for this project and related experiments. An illustration of a sharded collection can be found in Figure 38.

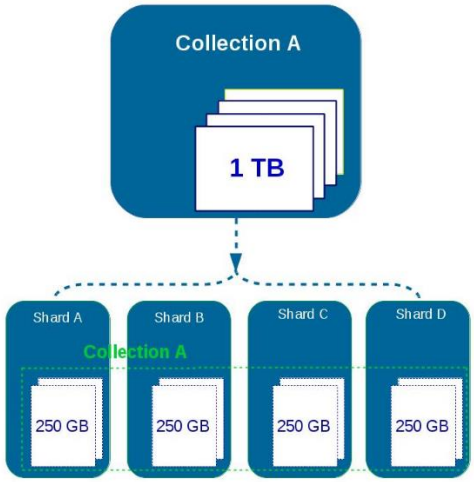


Figure 38: MongoDB Collection Shards (Maxim, 2016)

Using this storage format, a list of normalised dictionary results returned from IWS handlers could be saved by calling the 'insert_many' method provided by the MongoDB python library. This approach was replicated for similarly structured data returned by PDNS handlers. Unit and integration testing of all components developed until this stage was conducted successfully, with satisfactory performance results. Any additional logging messages and exception handling was implemented at this stage. Due to the results received from testing, bulk batching of database requests was considered instead of utilising the 'insert_many' method at this stage. It was concluded that such a complex feature would be excessive for

storing IWS data, due to the low quantity of records generally obtained from the sources. However, such a feature was deemed necessary for any operation performing a few thousand or more consecutive queries to MongoDB, such as updating large quantities of PDNS documents. An example of the code used to achieve such batching operations can be found within the PDNS enrichment code in Appendix F.

3.3.4 Internet-wide Scanning Data Merging

With the successful implementation of data handlers and associated storage methods, a key objective of Informant is to improve upon the past reviewed tools previously identified in the literature by merging several data sources. As discussed in the literature review, bringing together many sources rather than solely relying on one source of data, as done in past research, will drastically enhance the amount and accuracy of data gathered on an individual asset. Such additional data can improve the service banner analysis process, which would inevitably allow for more accurate CPE construction and, thus, better known-vulnerability assignment accuracy.

In accordance with the design, an iterative approach was implemented to merge IWS data. The implemented approach considers two sources simultaneously, setting and merging attributes of documents with matching IP addresses using one of three designed approaches as outlined in Table 8. The attributes 'source', 'ip', and 'timestamp' fields were the only fields not required to be merged, with the 'ip', and 'timestamp' attributes being assigned values based on the last record merged per unique IP address. The source attribute was statically set to 'merged' at the end of each complete unique IP merge. The core logic responsible for the iterative approach of merging two sources can be found in Figure 39.

```
def merge(ipList, source1, source2):
    FinalResult: list = []
    try:
        # Merge (only matching records)
        for ip in ipList:
            for rec in source1:
                if (rec['ip'] == ip):
                    for rec2 in source2:
                        if (rec2['ip'] == ip):
                            FinalResult.append(mergeData(rec, rec2))

        # Add left over records from data sources
        for ip in ipList:
            if not any(item for item in FinalResult if item['ip'] == ip):
                for rec in source2:
                    if ip in rec.values():
                        FinalResult.append(rec)
                for rec in source1:
                    if ip in rec.values():
                        FinalResult.append(rec)
    except Exception as e:
        print(f'[INFORMANT] Critical merging error. Please report: {e}')
```

Figure 39: Core IWS Data Merging Logic

The function in Figure 39 accepts two sources in addition to a list containing all unique IP addresses retrieved by all enabled data sources. The first logical set of nested for loops is responsible for merging and initialising the attribute merging process if matching IP addresses are identified within both datasets. The second set of nested for loops is for finalising the merging of two sources by duplicating data on IP addresses that do not occur in both sources. This is required to ensure that data not detected by multiple sources is not abandoned. As the amount of sources is defined by the end user, the function in Figure 39

is called several times to ensure all sources are merged regardless of the users' selection of sources, see Figure 40.

```
mergedResults = merge(ipList, shodanResults, censysResults)
mergedResults = merge(ipList, mergedResults, beResults)
mergedResults = merge(ipList, mergedResults, onypheResults)
```

Figure 40: Recursive Merge Function

The process of attribute merging follows one of three previously designed approaches noted below:

1. Selection of most reoccurring value. If multiple, the initial source value is selected.
2. Combining values and discarding duplicates.
3. Combining values and discarding duplicates based on the most recent timestamp data available.

Approach 1 is the most relied on approach and is performed on attributes that are commonly known to only have one individual static value and are likely to have a consistent result across each IWS platform. These attributes include asn, org, country, os and os_distro. For each of these attribute values, a collection is created containing the data held within all available IWS data gathered. Each list is then checked for the most common reoccurring element, which is selected as the merge value. The algorithm is O(n), where n is equal to the length of elements. In the event that the most common recurring element cannot be conclusively determined, the first initial source value is selected as the dominant value for the merge. Ideally, in such an event of an inconclusive result, the IWS platform with the latest data would be selected; however, this approach was not adopted due to free account limitations and project time constraints.

The second approach implemented caters towards attributes commonly associated with multiple values, allowing for differing sets of data gathered by multiple IWS platforms to be merged efficiently into one dataset. The attributes merged this way are hostnames, domains, ports and protocols—all attributes which can and often do contain more than one value. The approach combines the sources into a list set during the merging process in order to remove duplicate values. This function is called recursively for each source merged. An example of the practical hostname implementation can be seen in Figure 41, with each mentioned attribute passing through a similar procedure. This adds a significant amount of unnecessary duplicate code, which theoretically could be vastly improved. However, having each merge attribute separate is paramount to allow for easy debugging and attribute specific exceptions to be added efficiently during the experimentation phase.

```
try:
    # Combine and drop dupes from hostnames
    hostnames = list(set(source1.get('hostnames', [0]) + source2.get('hostnames', [0])))
    try:
        # Remove default hostname value is applicable
        hostnames.remove(0)
    except Exception:
        pass
except Exception:
    hostnames: list = []
    print("Critical hostnames merging error occured, hostnames data may be unreliable. Try using a different combination of Internet-wide scanners.")
```

Figure 41: Hostname Merge

Merging this specific attribute will help address issues encountered by past researchers concerning the lack of service banner information obtainable (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019). To understand how this process was implemented and works, a more profound grasp of banner

data retrieved from IWS handlers must be obtained. The structure of such data can be observed in Figure 42.

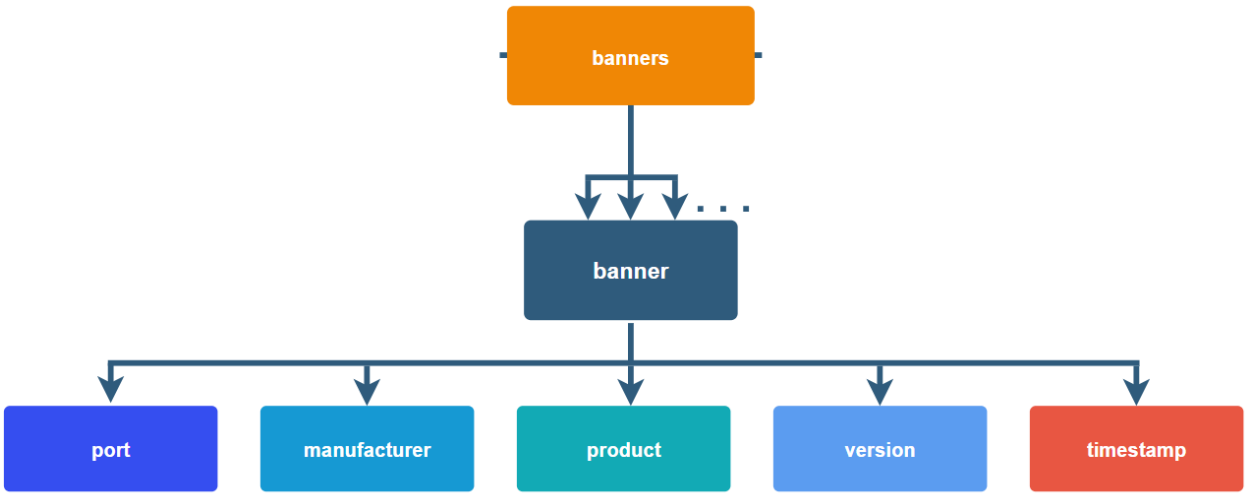


Figure 42: IWS Banner Data Structure Breakdown

The five sub-attributes shown in Figure 42 can potentially be contained within each service banner retrieved by IWS handlers. The goal of the merging process is to determine the most accurate banner information available, which is achieved by considering the timestamp attribute associated with the time each IWS platform last detected the banner. This data is stored within the timestamp sub-attribute of each service banner retrieved. As shown in Table 9 below, Onyphe is the only platform that fails to provide service specific last seen timestamp information. Therefore, it is manually appended a time of 00:00:00, the earliest possible scan time.

Table 9: Retrieved IWS Platform Banner Timestamp Formants

IWS Platform	Banner Timestamp Format
Shodan	YEAR-DATE-DAY H:M:S
Censys	YEAR-DATE-DAY H:M:S
BinaryEdge	YEAR-DATE-DAY H:M:S
Onyphe	YEAR-DATE-DAY

The banner merging process is performed during the core recursive merge function – see Figure 40 - and refined after all recursive merging processes have been completed. Within the recursive merging function previously discussed, banner data gathered from all selected sources is combined into one list. Upon completion of all merging, the data within the banner section of each merged asset is refined based on the banner port number and associated timestamps. In the case of duplicate service banners being detected – based on port numbers, the most recently detected banner would be kept, and alternative banners on the same port would be discarded. This approach ensures the most recent up to date service banners are always selected whilst maintaining the ability to produce an extensive list of services out with the scope of one IWS platform's capabilities. The algorithm performing this task sorts the merged banners in port order before utilising a lambda function to determine the most recently retrieved duplicate service banners. This aspect of the algorithm can be found in Figure 43.

```python
def bannerMerge(banners):

    # Function to remove duplicate banner information based on timestamp
    bannersSorted = sorted(banners, key=itemgetter('port'))

    result = collections.defaultdict(list)
    for d in bannersSorted:
        result[d['port']].append(d)
    result_list = list(result.values())
    banners = []

    for item in result_list:
        tmp = []
        for value in item:
            tmp.append(value)
        tmp = sorted(tmp, key=lambda tmp: tmp['timestamp'])
        banners.append(tmp[-1])

    return banners
```

Figure 43: Service Banner Merging Algorithm

Upon unit testing of all implemented merging features, rigorous integration testing was performed to validate the entire merging process from start to finish. A purposely setup device exposed to the internet was set up with services continuously being manipulated over a one-week period to test all aspects of the merging code thoroughly. Using this device as a test bed, several checks were conducted via the CLI using the implementation of Informant up until this point. All aspects of the merging process were found to be nominal. The final merged record can be seen in Figure 44.


```
source: "merged"
ip: "91.
domains: Array
hostnames: Array
org: "UAB Technova LT"
asn: "AS42675"
country: "United Kingdom"
ports: Array
  0: 80
  1: 22
protocols: Array
  0: "22/ssh"
banners: Array
  0: Object
    port: 22
    product: "OpenSSH"
    version: "7.6p1 Ubuntu 4ubuntu0.3"
    timestamp: "2021-03-17 10:16:41"
  1: Object
    port: 80
    product: "Apache httpd"
    version: "2.4.29"
    timestamp: "2021-03-22 17:49:24"
os: Array
  0: "Linux"
os_distro: Array
  0: "Ubuntu"
timestamp: 2021-03-23T14:00:00.000+00:00
```


Figure 44: Example of Merged Record

Breaking down and analysing the most crucial aspect of the merging process, banner merging, the effectiveness of the implementation can be determined. To do this, information gathered by all four IWS platforms was compared against the known device service configuration at the time. The known configuration can be seen in Figure 45.

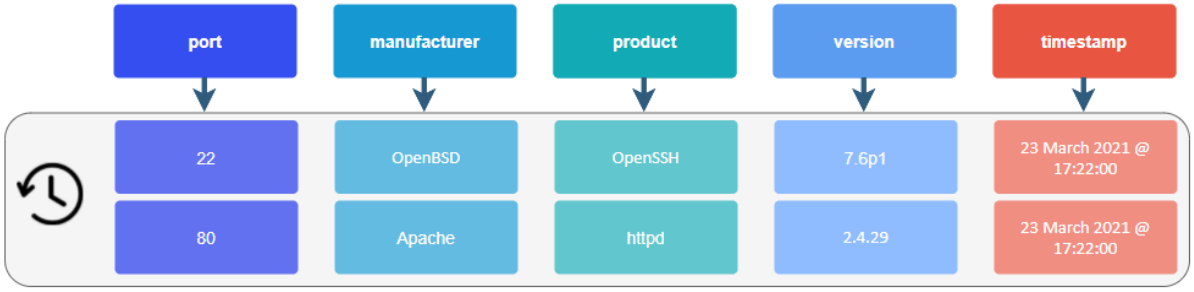


Figure 45: Realtime Test Server Service Configuration

Using the data in Figure 45 as a baseline and conducting a comparison with the merged data constructed from all available IWS sources in Figure 46. It was concluded that the adopted approach was highly effective. The banners selected for the merge, in this case, were from two sources, BinaryEdge and Shodan. Referring back to the literature review, as the ports are known to be scanned by both platforms that adopt a similar scanning methodology, the resulting selection of platforms is likely solely down to the differing service scanning frequencies of the platforms (Li *et al.,* 2021).

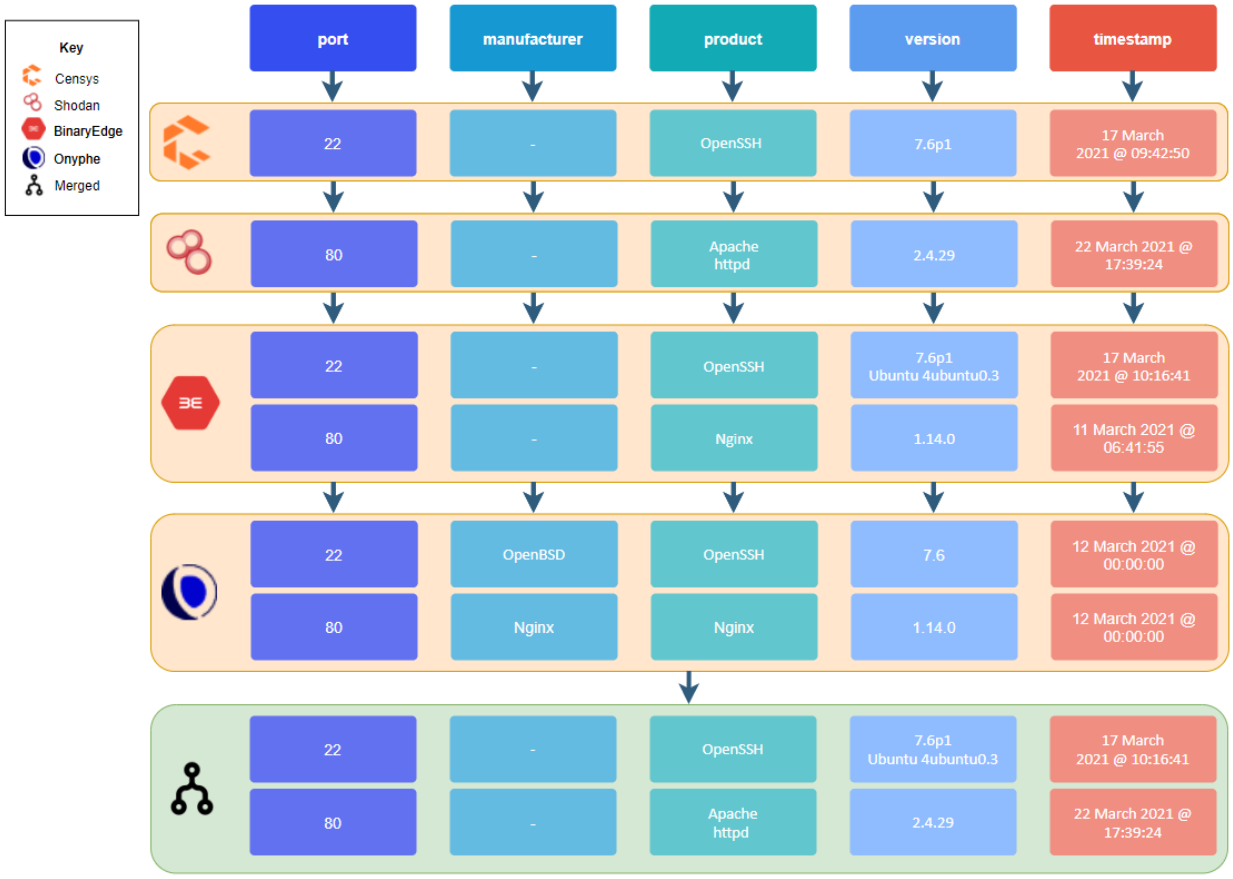


Figure 46: Test Banner Merge Breakdown

Further analysing the results in Figure 46, it can be concluded that the implemented merging algorithm addresses past issues identified by researchers relating to scanning delays on specific platforms. It achieves this by selecting the most current service data corresponding to each unique port identified from

the available selection of IWS platforms. In this test case, the IWS handlers were queried on the 23rd of March at 15:00 hours. As can be observed from the results the latest data available is provided by Shodan and was gathered on the 22nd of March at 10:16 hours. Over 24 hours prior to the data being requested, showing that scanning delays, although severely mitigated by utilising multiple data feeds, can still be greater than 24 hours. Without significant advances in IWS techniques, this will remain the case. The only current practical way of resolving this issue is by running on-demand scans, which is often credit intense and a restricted feature of IWS platforms. However, this unit test made it clear that in this scenario, Informant would vastly outperform both ShoVAT and Scout when it comes to gathering accurate banner information (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019). This can be concluded as information from both Shodan and Censys, the two platforms used by ShoVAT and Scout was inconclusive, with each platform missing at least one service during scans. Informants ability to use multiple data sources allowed data from Censys and BinaryEdge to construct a more accurate picture of the asset.

3.3.5 CPE Construction

In this section, service banner information gathered and merged from the users' selection of IWS platforms is analysed in an attempt to reproduce and associate accurate CPE identifiers. The approach in the design phase based on reviewed literature was implemented following the flowchart in Figure 20.

The preliminary design conceived from past work by O'Hare, Macfarlane and Lo (2019) was adapted to accept separate attributes, as obtained by the IWS handlers. In addition to the inclusion of an exception operation to help further mitigate tedious issues surrounding depreciated CPE values and varying banner inconsistencies introduced by using multiple IWS platforms. The first stage in the process was to combine all known service banner attributes into one string based upon the official CPE specification (FIRST, 2019) . This process is shown in Figure 47. Once complete, validation of the formed CPE could commence using the Levenshtein algorithm.

```
jStr = ':'
cpeArray = ['cpe:2.3', 'a', manufacturer, product, version, '*','*', '*', '*', '*', '*', '*', '*']
p = jStr.join(cpeArray).lower()
cpeString = exceptions(cpeArray[2].lower(), vendor.lower() ,cpeArray[3].lower(), cpeArray[4].lower())
```

Figure 47: Service Attributes to CPE String

To replicate the Levenshtein algorithm used within the previously produced Scout tool in a time efficient manner, the performance focused RapidFuzz Python library utilising the Levenshtein distance algorithm to perform string matching was implemented. RapidFuzz was selected over the more well-established fuzzywuzzy Python library due to its C based nature and exponential increase in performance (Bachmann, 2021). Several differing ratio scoring methods are available within the library, each using varying techniques, with each technique specialising in specific use cases for string matching. The available ratio methods within the library can be observed in Table 10. Analysing the source of the Scout tool, it was identified that a simplistic ratio calculation was implemented, comparable to the simple ratio method found within Table 10. This basic approach was based on the less able editdistance Python library (Tanaka, 2019). To improve upon this previous approach, the capabilities of each ratio method made available by the selected library were analysed to determine if any improvement was possible.

Table 10: Ratios for Levenshtein Distance Algorithm

Ratio Method	Description
Simple Ratio	Basic measurement of edit distance between strings. Best suited for very short and very long strings.
Partial Ratio	Measurement of edit distance between substrings. Best suited for partial string matches.
Token Set Ratio	Attempts to rule out string differences, measurement of edit distance called on three substring sets. Best suit for determining how similar and different strings are.
Token Sort Ratio	Measurement of edit distance between strings out of order. Best suited for string matches in out of order context.
Partial Token Set Ratio	Measurement of edit distance between words in the strings based on unique and common words between them.
Partial Token Sort Ratio	Measurement of edit distance between substrings out of order. Best suited for partial string matches in out of order context.
Weighted Ratio	Utilises the simple ratio method, and if deemed necessary based on the inputted string various other ratio methods are run and the highest matching result(s) returned.
Quick Ratio	Basic measurement of edit distance between strings, with some validation and short-circuiting.

Based on the comparison of ratios in Table 10, the weighted ratio method was deemed as the best approach to adopt. Using this ratio method, the 'extractOne' method with a cut-off threshold of 90% could be relied on with confidence to provide accurate CPE validation checks. Figure 48 shows how test data on port 22 obtained from Onyphe results in Figure 46 can be used to produce a valid CPE using this method accurately. The core source code for the CPE construction process can be seen in Appendix C.

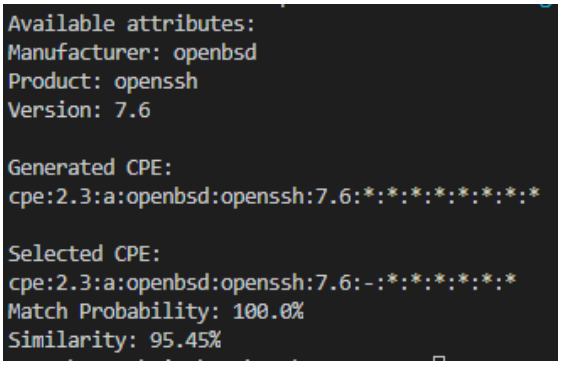


Figure 48: Accurate CPE Generation

As can be seen in Figure 48, it is possible to validate precise CPEs using this method. In this case, the true CPE value is in fact 'cpe:2.3:a:openbsd:openssh:7.6:-:*:*:*:*:*:*' which has a 95.54% similarity with the original generated CPE of 'cpe:2.3:a:openbsd:openssh:7.6:*:*:*:*:*:*:*:*'. The match probability is 100% and is derived from the weighted ratio method utilised. However, even with the additional layer of ratio complexity, issues identified by Scout were still discovered to be present when attempting to match CPEs containing popular words accurately.

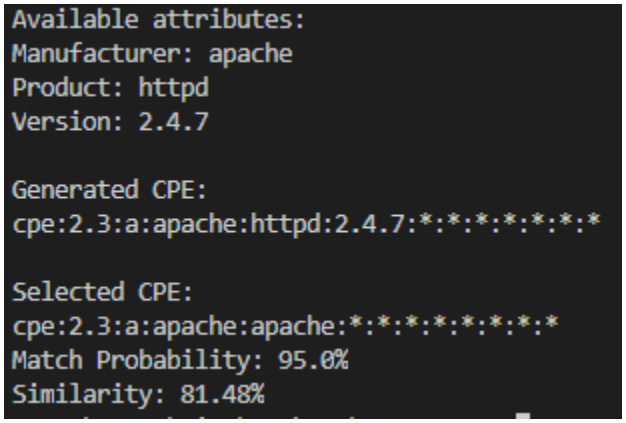


Figure 49: CPE Validation Issues

Figure 49 shows such an issue. As can be seen, the manufacturer, product and version are all present. Although the final selected CPE corresponds to the correct service, the version information is not included in the final selected CPE. This issue has the potential to significantly affect results negatively during the experiment stage of this project. To address this issue, a set of vendor specific exceptions were created, as shown in Appendix D. Additionally, all depreciated CPEs were removed from the NVD data to increase the chance of an exact match. Upon implementing these measures, the accuracy of the CPE verification process was significantly improved. Using the merged service banner data in Figure 46, both corresponding CPE values were correctly identified, as shown in Figure 50. Successfully mitigating the previously discovered issue.

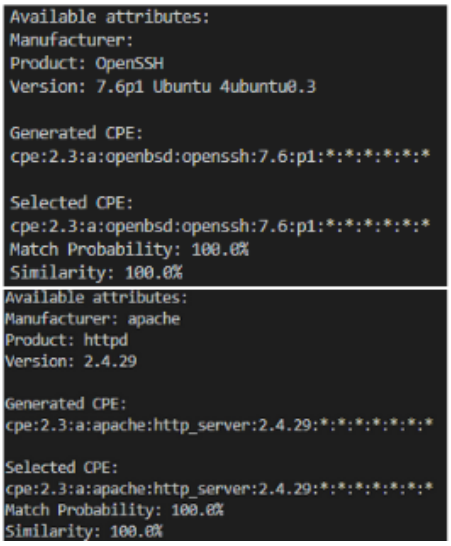


Figure 50: CPE Validation

Upon completing the CPE validation process, the identified verified CPE is looked up and assigned associated CVEs. This process is achievable as the cve-search dependency used to obtain data from the NVD contains a correlated list of all potential CPEs corresponding to each CVE document. Within this dataset, using the data stored under the 'vulnerable_configuration' field, CVE IDs and associated vulnerability data for each can be retrieved from the vulnerability database by searching for validated CPEs. If version information is not available, the accuracy of CPE validation is significantly diminished. Therefore, the CVE assignment process is skipped assuming the ratio threshold is not met, resulting in the service banner being classified as obfuscated. This is done to mitigate false positives, as without version information, it is not straightforward to determine if a service is vulnerable accurately. Figure 51 displays the code used to perform this process, with a further breakdown of the code available in Appendix E.

```
def cve_lookup(cpe, version, service, score) :
    # Find known vulnerabilties matching vulnerable config
    cves = vulDB['cves'].find({"vulnerable_configuration" : cpe})
    cveList = []

    if (cves.count() > 0 and version != '') :
        print ("Vulnerabilities found:")
        for cve in cves :
            print("CVE: " + str(cve["id"]))
            cveFound = {'cve' : cve["id"], 'cvss2': cve["cvss"], 'vector' : cve["cvss-vector"], 'cpe' : cpe, "port": '', "manufacturer": '', "product": '', "version": '', "cpe_score": ''}
            cveFound['port'] = service.get('port', '')
            cveFound['manufacturer'] = service.get('manufacturer', '')
            cveFound['product'] = service.get('product', '')
            cveFound['version'] = service.get('version', '')
            try:
                cveFound['cpe_score'] = score
            except:
                cveFound['cpe_score'] = 'Not found'

            cveList.append(cveFound)

        return cveList
    elif (version == '') :
        return 'obfuscated'
    else:
        print ("No vulnerabilities found.")
        return None
```

Figure 51: CVE Assignment

3.3.6 Passive DNS Enrichment

As previously discussed in the design phase, to take advantage of ingested PDNS data for risk assessment, further data enrichment is required. Each PDNS record requires the ASN to be obtained, and if feasible, the netname. Two approaches were implemented to obtain this additional information. One for large bulk queries, utilising Team Cymru's excellent API for mapping IPs to their corresponding ASN (Team-Cymru, 2021). Another for facilitating small queries - sub-fifty address ASN lookups – which employ the Registration Data Access Protocol (RDAP) for lookups. Both approaches allow for ASNs and netnames to be obtained easily.

In order to not flood the generous free service designed for large bulk requests provided by Team Cymru, if the quantity of lookups required is no more than fifty IP addresses, then enrichment data is fetched from the associated internet registrar via the RDAP. The code to perform this operation can be found in Figure 52. Furthermore, retrieved data was saved to the associated documents containing PDNS record data iteratively.

```
print("[INFORMANT] Using Standard ASN & Netname Lookup")
for record in mainDB[project_name].find({'timestamp': timestamp}):
    if 'type' in record:
        if (not record['ip'].islower() and not record['ip'].isupper()):
            try:
                lookup = IPWhois(record['ip']).lookup_rdap(asn_methods=['dns', 'whois', 'http'])
                mainDB[project_name].find_one_and_update({"_id": record['_id']},{"$set": {"asn": lookup['asn']}})
                mainDB[project_name].find_one_and_update({"_id": record['_id']},{"$set": {"netname": lookup['network']['name']}})
            except Exception as e:
                print("[INFORMANT] PDNS ASN & Netname Enrichment Error (Standard method): " + str(e))
                continue
```

Figure 52: RDAP Approach to ASN and Netname Lookup

During testing using the adopted RDAP approach, fifty requests were found to be the maximum number of requests that could be processed before issues were encountered. If exceeded, especially by a drastic amount, the time taken to complete the process increased drastically. Therefore, large bulk lookup requests employed the Team Cymru approach. In order to perform bulk queries using the Team Cymru API, pre-processing of IP data was required to ensure the API would accept it.

```
begin
verbose
68.22.187.5 2005-06-30 05:05:05 GMT
207.229.165.18 2005-06-30 05:05:05 GMT
...
198.6.1.65 2005-06-30 05:05:05 GMT
end
```

Figure 53: Required Bulk ASN Lookup Format (Team-Cymru, 2021)

Once the IP data was in the required format, as shown in Figure 53. The data could be converted into bytes and sent via the netcat protocol to the Team Cymru API, awaiting a response. See Figure 54.

```
def cymru_query(bulk_query, timeout):
    # Team Cymru IP to ASN Query (netcat)
    try:
        data = ""
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(timeout)
        s.connect(("whois.cymru.com", 43))
        s.sendall(bulk_query)
        reply = s.recv(4098)
        data = reply
        # Gets data until an empty line is found.
        while True:
            reply = s.recv(1024)
            data += reply
    except socket.timeout:
        if data != '':
            pass
        else:
            raise
    except Exception as e:
        raise e
    finally:
        s.close()

    return data
```

Figure 54: Team Cymru Query Code

Upon retrieval of a response from the API, the returned data can be refined using a series of string manipulation methods to extract the desired information. The extracted data can then be used to enrich the PDNS records within the MongoDB based on the IP field, with multiple PDNS records potentially corresponding to one IP address – one to many relationship. However, as mentioned in section 3.3.3,

this process is likely to result in thousands of operations. During unit testing a significant performance bottleneck was identified when saving large quantities of data via standard update methods. Further research revealed that the cuprite was due to several MongoDB API requests being made per update call. Therefore, a batching approach needed to be implemented to achieve acceptable performance levels when interfacing with MongoDB. This issue is specifically prominent in this case as the underlying architecture of the update many documents operation consists of multiple queries. The adopted batched 'UpdateMany' approach and data extraction method can be seen in Figure 55.

```
bulk_updates = []
for entry in string.splitlines():
    try:
        split_entry = entry.split("|", 6)
        asn = split_entry[0].strip()
        ip = split_entry[1].strip()
        netname = split_entry[6].strip()
        netname = netname.split(",", 1)
        bulk_updates.append(UpdateMany({"$and": [{"ip": ip}, {'timestamp': timestamp}]}, {'$set': {'asn': asn}}))
        bulk_updates.append(UpdateMany({"$and": [{"ip": ip}, {'timestamp': timestamp}]}, {'$set': {'netname': netname[0]}}))
    except Exception as e:
        print(str(e))
            continue

mainDB[project_name].bulk_write(bulk_updates)
```

Figure 55: Bulk ASN & Netname Lookup Refinement and Batching

Using the enriched documents, simple logical comparison checks were run to check if each ASN name detected matched that of the CIDR or IP address originally scanned. If not, the PDNS record would be flagged as an external asset that can undergo further manual review to determine if the flagged PDNS record is dangling. A large segment of the PDNS enrichment code can be found in Appendix F.

3.3.7 Default Vulnerabilities, rDDoS and Port Assessment

In an attempt to analyse and highlight CVEs within the NVD that are exploitable based on their default service configuration, the second proposed approach within the design section was implemented. As a result, the attack complexity and attack vector components from within the CVE vector were considered based on information obtained from the CVE specification (FIRST, 2019). The individual components of interest that make up each CVE can be seen below.

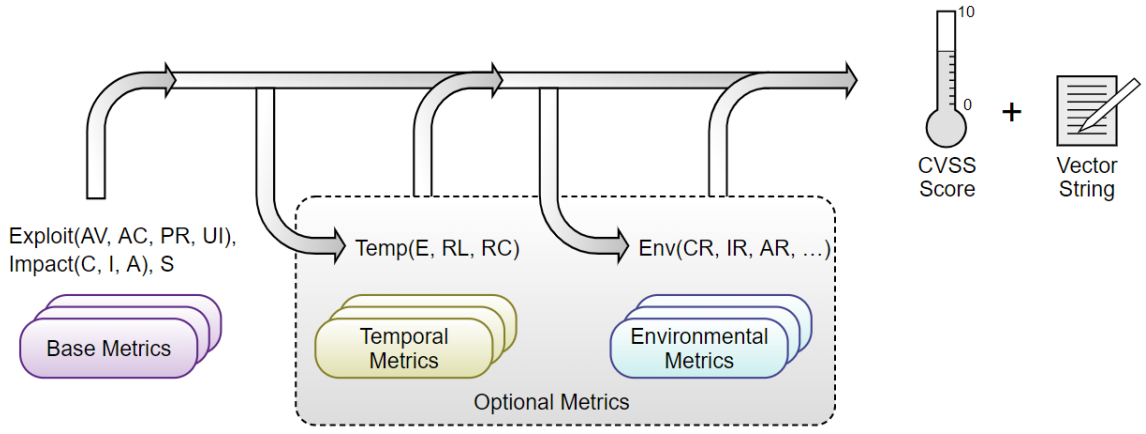


Figure 56: CVSS Metrics and Equations (FIRST, 2019)

The attack vector was first refined to network only based attacks as denoted by 'AV:N' and then further refined by low attack complexity, denoted by 'AC:L'. The code for this feature can be found in Appendix G. Although this is far from an adequate solution for indicating if a vulnerability is certainly vulnerable by default, it can be used as a rough indicator based on information obtained from the official specification (FIRST, 2019). Many additional attributes, notably the CVSS score and temporal scores, offer additional insight into determining this factor. However, few CVEs have temporal scores, and the consistency of CVSS scores has been questioned by past researchers (Anwar *et al.,* 2020).

Considering the mentioned drawbacks of the above implemented approach, the alternative NER based approach planned during the design phase could have been implemented to help improve the reliability of default config CVE detection mechanism. However, the project time frame did not allow for such a time-intensive feature to be fully implemented, leaving room to expand upon this aspect of the project in the future. In such an event, spaCy (2021), a high-performance off-the-shelf ML model, would most likely be adopted for the approach to ensure minimal additional performance overhead. A custom classification and labelling system to determine if a CVE was exploitable by default would then be implemented, and training data produced. All data labelling would solely be based on the context of CVE summaries. Experimenting briefly – see Appendix H - with this approach highlighted that not enough information was available within the vast majority of CVE summaries to determine such a factor accurately. Related issues with attempting to classify data via similar means have been elaborated on by past research found in literature. Guo, Xing and Li (2020) found that only 3% of CVE summaries describe both vulnerability type and cause, meaning that the likelihood of finding useful data relating to if the vulnerability is default or not is unlikely. However, the potential of combining the fully implemented vector approach and a text-based mining NLP based approaches to detect default config vulnerabilities is a promising future research avenue. A whole paper could be written on this topic alone, hence the need for further research in this area.

The predicted rDDoS throughput calculation by Leverett and Kaplan (2017) was implemented by manually translating the BAF data shown in Figure 22 into a machine-readable JSON format. A function was then created to compare this data against IWS port information to check if the port was known to operate a service with rDDoS potential. If so, the sum of the maximum BAF for the services detected was calculated, and the total numerical value was assigned to the asset as a risk using the same efficient batched update approach utilised by the PDNS enrichment task. This operation is called upon a risk assessment being run, a task operation that the user can initiate. Associated code for this feature can be found in Appendix I.

The port risk assessment feature is the most simplistic feature of all to be implemented. Each exposed port detected by IWS data is compared against a list of user-defined ports contained within the core MongoDB database. The matching data is flagged as a risk within the corresponding merged asset document if a match is detected. This operation is also only callable upon a risk assessment being run.

3.3.8 Output

As highlighted in the design, a minimalistic web interface was developed to allow for a comprehensive and friendly end user experience. From the wireframe in Figure 23, the dashboard observed in Figure 57 was designed and integrated into the backend.

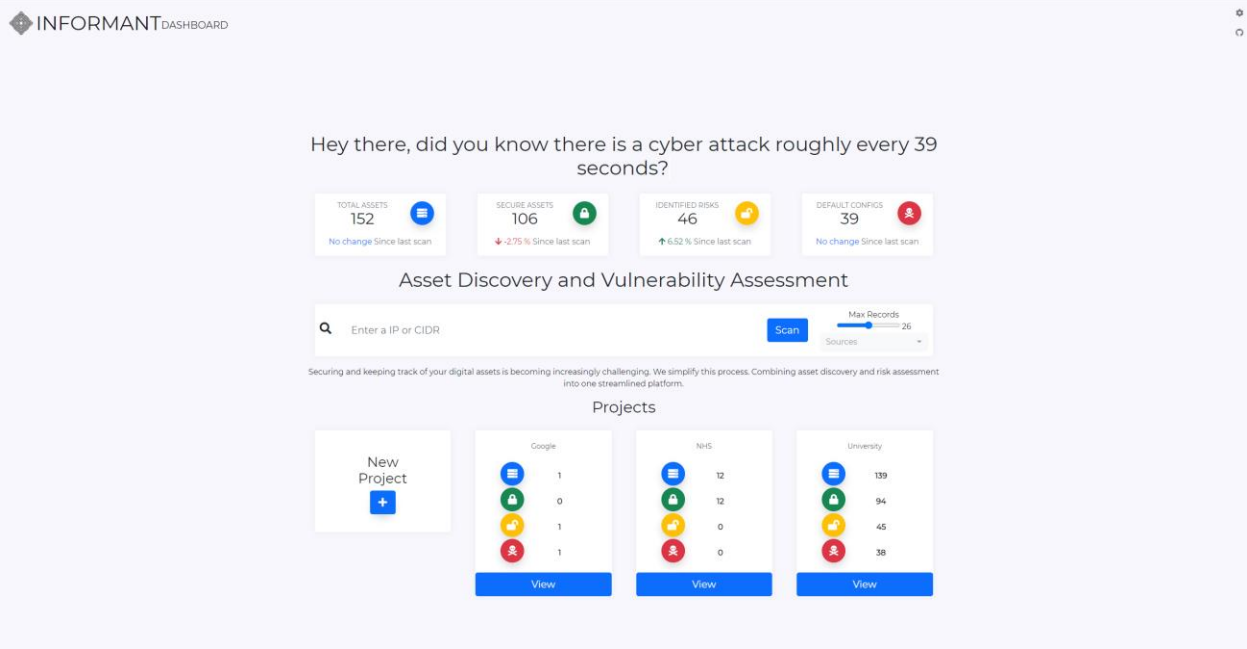


Figure 57: Informant Overview Page

Keeping the design style consistent throughout, the settings, overview and several data drilldown views were developed. All of which can be found in Appendix J. The implementation of the server-side data tables, which are present on a multitude of pages, is built around the jQuery DataTables library (DataTables, 2021), with data processing conducted server-side using Python. A simplified diagram representing how this process works can be observed in Figure 58, alongside a sample Ajax response for a paginated data table in Figure 59.

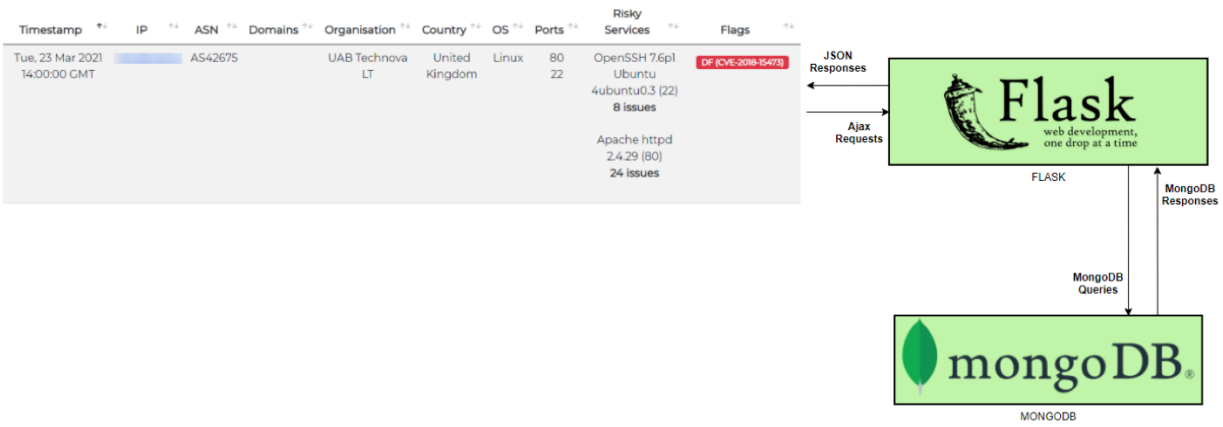


Figure 58: Data tables Architecture Overview

```
127.0.0.1 - - [25/Mar/2021 15:59:57] "GET /tables/IWS_0/University/21/03/2021,22/03/2021?sEcho=4&iColumns=10&sColumns=%2C%2C%2C%2C%2C%2C%2C%2C%2C%2C&iDisplayStart=0&iDisplayLength=10&mDataProp_0=timestamp&sSearch_0=&bRegex_0=false&bSearchable_0=true&bSortable_0=true&mDataProp_1=IP&sSearch_1=&bRegex_1=false&bSearchable_1=true&bSortable_1=true&mDataProp_2=ASN&sSearch_2=&bRegex_2=false&bSearchable_2=true&bSortable_2=true&mDataProp_3=Domains&sSearch_3=&bRegex_3=false&bSearchable_3=true&bSortable_3=true&mDataProp_4=Organisation&sSearch_4=&bRegex_4=false&bSearchable_4=true&bSortable_4=true&mDataProp_5=Country&sSearch_5=&bRegex_5=false&bSearchable_5=true&bSortable_5=true&mDataProp_6=OS&sSearch_6=&bRegex_6=false&bSearchable_6=true&bSortable_6=true&mDataProp_7=Ports&sSearch_7=&bRegex_7=false&bSearchable_7=true&bSortable_7=true&mDataProp_8=Services&sSearch_8=&bRegex_8=false&bSearchable_8=true&bSortable_8=true&mDataProp_9=Flags&sSearch_9=&bRegex_9=false&bSearchable_9=true&bSortable_9=true&sSearch=21&bRegex=false&iSortCol_0=7&sSortDir_0=asc&iSortingCols=1&_=1616687988590 HTTP/1.1" 200 -
```

Figure 59: Data tables Ajax Response

In order to effectively bring critical information to the attention of end users at a glance, a series of colour-coded risk flags were developed to catch the users' attention. Each flag and corresponding description can be found in Table 11.

Table 11: Informant Flag Declarations

Flag	Dataset	Description
Obfuscated Service (80)	IWS	The Obfuscated Service flag is risen if the version number of a service cannot be derived. The numerical value encapsulated within the brackets represents the affected port.
DF (CVE-2010-1452)	IWS	The Default Config (DF) flag is risen if a CVE is detected as being exploitable by default. The encapsulated text within the brackets represents the corresponding CVE and is interactable.
rDDoS Potential (54)	ISW	The rDDoS Potential flag is risen if a service is suspected to be susceptible to rDDoS. The numerical value encapsulated within the brackets represents the maximum total BAF.
High Risk Ports (22)	ISW	The High-Risk Ports flag is risen if a port contained on the high-risk port list in detected on a given asset. The numerical value encapsulated within the brackets represents the affected port.
External ASN (Review)	PDNS	The External ASN flag is risen upon DNS records not corresponding to the primary ASN from the initial scan range.

An automatically generating summary page for each user created project was developed to show information in a user-friendly manner and ensure essential data was not misinterpreted. The page draws out crucial overall project stats, PDNS and IWS risks in one clean and concise location. All of which can be filtered and updated on the go via a draggable interactive timeline and search boxes, which utilise Ajax to ensure smooth transitions between time-series data across the entire page. Scanning and risk assessment options are also present on the page, as well as the inclusion of a near real-time up-to-date view of the Redis task queue. Additional extra features were added during usability testing to make the experience more user-friendly for the end user, including the integration of informative tooltips, project settings page, data export capabilities and the ability to delete a project. The project summary page can be found in Figure 60.

The Redis message broker and task queue were integrated during this phase. The IWS ingest and risk assessment functions both take advantage of this feature, allowing for the web front-end to be useable whilst lengthy tasks are running in the background. Also, if more than one background worker is available, this feature theoretically allows for multiple tasks to run concurrently. However, this aspect of concurrent tasks is currently not considered by the front-end design as it is not deemed essential. Snippets of associated code can be found in Appendix K.

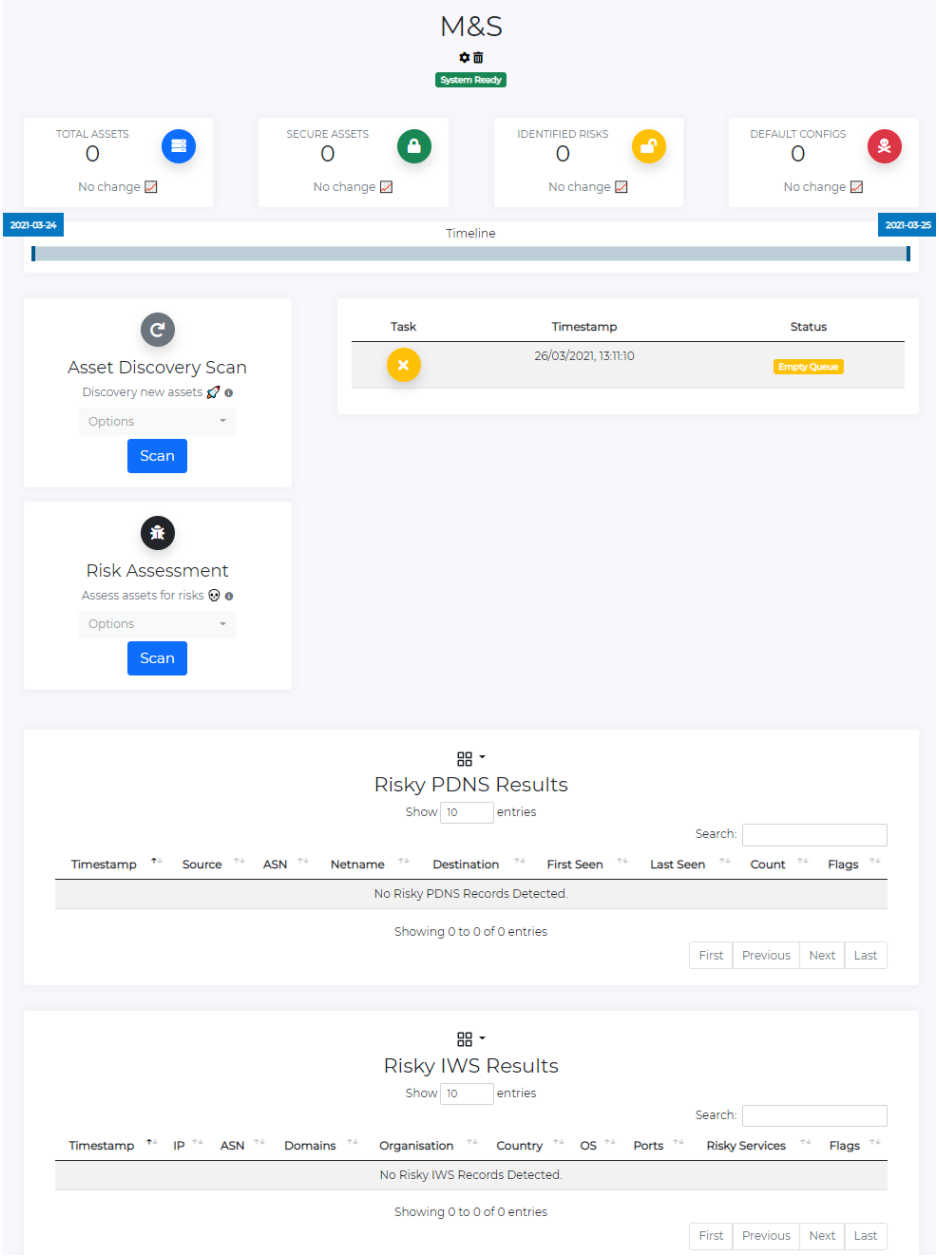
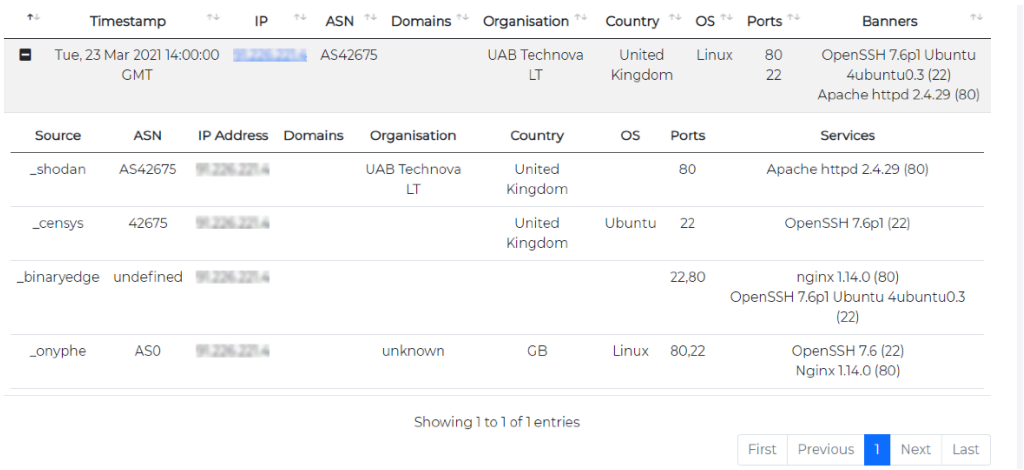


Figure 60: Project Summary Web Interface

Data drilldowns allowing for all PDNS and IWS data to be visually analysed were implemented separately using data tables. Of the two implementations, displaying and correlating all IWS data was the most challenging. The approach adopted involved grouping all associated IWS data retrieved with the merged asset and displaying the data within interactive dropdown tables within a master data table. The final product resulted in a very effective and transparent design, as shown in Figure 61. Additionally, a drilldown into all data stored on each individual detected asset is possible.



Timestamp	IP	ASN	Domains	Organisation	Country	OS	Ports	Banners
Tue, 23 Mar 2021 14:00:00 GMT	[redacted]	AS42675		UAB Technova LT	United Kingdom	Linux	80 22	OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (22) Apache httpd 2.4.29 (80)

Source	ASN	IP Address	Domains	Organisation	Country	OS	Ports	Services
_shodan	AS42675	[redacted]		UAB Technova LT	United Kingdom		80	Apache httpd 2.4.29 (80)
_censys	42675	[redacted]			United Kingdom	Ubuntu	22	OpenSSH 7.6p1 (22)
_binaryedge	undefined	[redacted]					22,80	nginx 1.14.0 (80) OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (22)
_onyphe	AS0	[redacted]		unknown	GB	Linux	80,22	OpenSSH 7.6 (22) Nginx 1.14.0 (80)

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

Figure 61: IWS Breakdown Web Interface

The final front-end element to be implemented was the integration of an interactive auto-generated node-based network visualisation. For this feature, all asset IP addresses are represented by cylinders, and all corresponding connected domains are represented by ovals, as shown in Figure 62. The visualisation allows for a clear picture of assets to be depicted easily, especially useful in scenarios where large complex projects are being investigated. PDNS data, domains and hostnames gathered from IWS data are used to construct the visualisation.

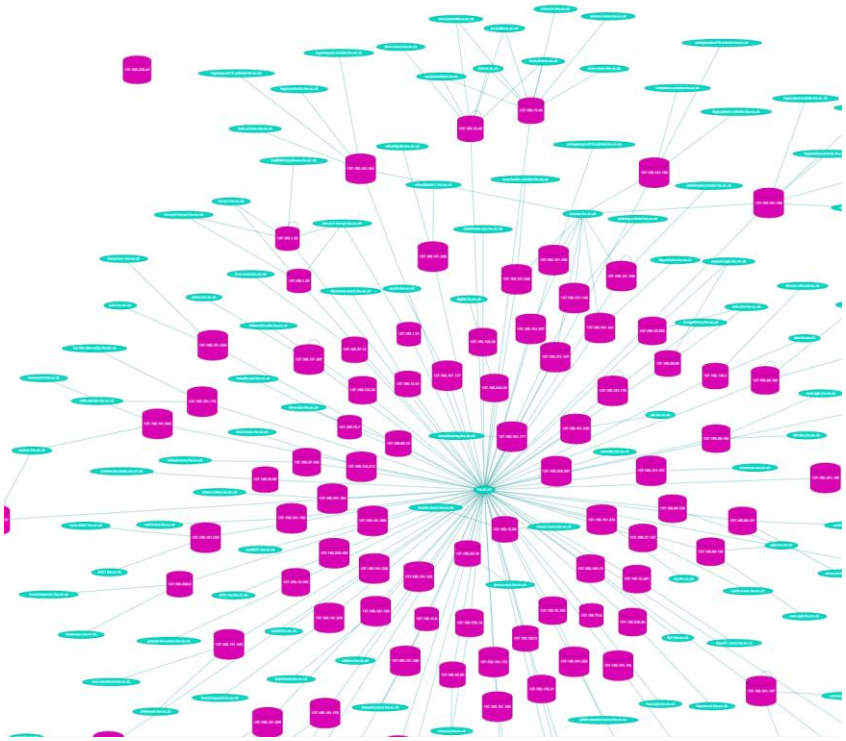


Figure 62: Automated Network Node Map

The final implemented solution is open-source and has been made publicly available via GitHub (LA-Shill, 2021). Alternatively, examples of all core UI components can be seen in Appendix J.

3.4 EXPERIMENTATION

The tool developed in this paper will be used to ingest and analyse large quantities of data from across four UK-based universities operating within the Janet network - one of the largest UK networks. IP allocations for each of the four UK universities are obtained by analysing associated Janet network Border Gateway Protocol (BGP) routes to determine the IP allocations of individual universities. For these experiments, two universities based in England and another two in Scotland will be selected. Universities were selected as ideal candidates for scoping out and verifying the functionality of Informant due to their extensive online presence, especially in the wake of the global Covid-19 pandemic. Additionally, the cybersecurity of such educational institutions is of particular relevance currently due to a series of high-profile attacks (Cimpanu, 2020; Hellard, 2020; Marzouk, 2021). Educational institutions are also publicly available. Hence ethical concerns are mitigated, with multiple pieces of related literature already provided insights into the security of educational institutions (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019). Focusing this tool against academia will give a true insight into how well the improvements to the CAR process have been in further exposing the security posture of educational institutions.

All developed data gathering components of the Informant tool follow the CAR process, resulting in no direct active connection being made to the designated target at any point. Similarly, to other CAR tools in literature, this allows for enumeration and risk assessment of systems to be conducted without revealing the host systems digital identity to the target, which is unavoidable using traditional active scanning-based approaches. The CAR process primarily achieves this by accessing freely available data provided by IWS platforms already continuously scanning the internet using various methods. Figure 63 depicts the difference between how Informant obtains data on targets in comparisons to OpenVAS, a similar tool utilising traditional active scanning techniques. The implementation of the geolocation verification feature is the only active component of the developed tool and will be fully disabled during experiments.

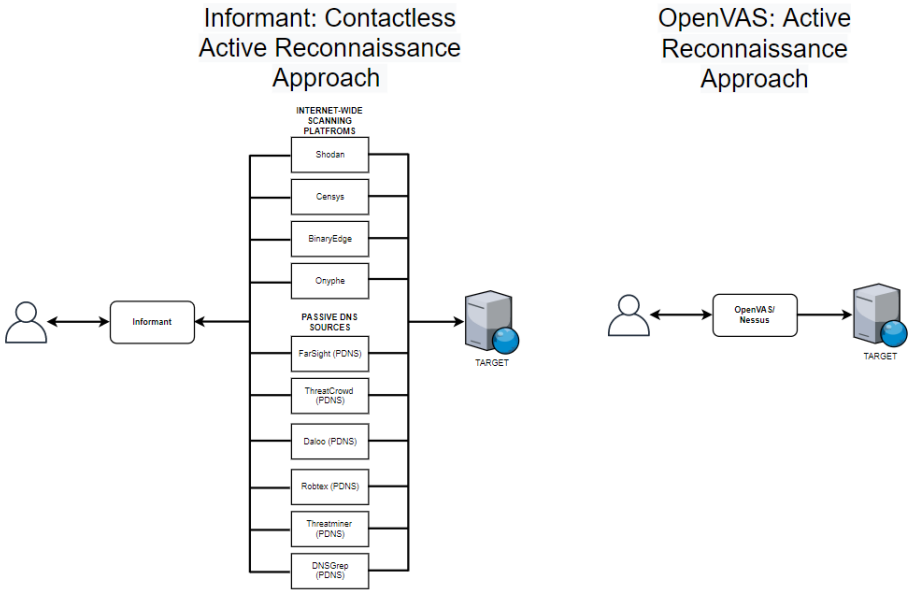


Figure 63: Contactless 'Active' Reconnaissance vs Active Reconnaissance

The below subsections provide an in-depth overview of all experiments planned. The results for each can be found within the corresponding section within Chapter 4.

3.4.1 Example Operation

Upon completion and integration of all implemented components, Informant is to be put to the test in a real-world scenario to ensure all non-active aspects of the tool meet the project objectives. This example operation will act as a real-world use case scenario for the tool, ensuring all features work in tandem and results are clearly shown via the user interface. A mediocre sized subsection of the Abertay University network was selected as the designated target for this initial experiment. All integrated IWS platforms and PDNS sources will be utilised for the operation, and a manual realistic high-risk port list will be set to mimic an end user, containing the following ports: 22, 23, 4786, 7547. The exact project settings for the experiment can be observed in Figure 64. The asset discovery and risk assessment aspects of the project will be of most interest during this experiment.

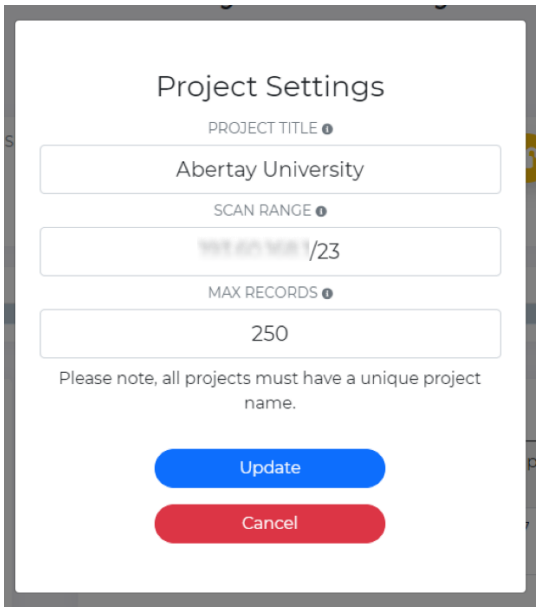


Figure 64: Example Operation Project Settings - Abertay University

To further evaluate the effectiveness and performance of Informant on a larger scale, four additional anonymised UK-based universities will also be analysed and compared during this stage. Aspects of these results will be briefly compared to results returned by ShoVAT and Scout.

3.4.2 Experiment 1 – Internet-wide Scanning Data Merging

The literature reviewed shows that no attempts to automate the merging of data obtained from multiple IWS platforms had been attempted. As such, a reproducible procedure for evaluating such a feature does not exist at present. Due to this, no comparative assessment could take place. Therefore, a manual assessment will be used to provide a fair analysis. This will be achieved using a series of five deployed UK-based servers as a small sample set, manually merging data obtained from IWS platforms and comparing the results against the automated merging process. Additionally, these results will be further compared against the actual server configurations to determine the accuracy of the overall merging process. The results produced will be analysed and compared to evaluate the effectiveness of this approach in

mitigating known IWS issues (Bodenheim *et al.,* 2014; Durumeric *et al.,* 2015; Li *et al.,* 2021). The experiment is illustrated in Figure 65.

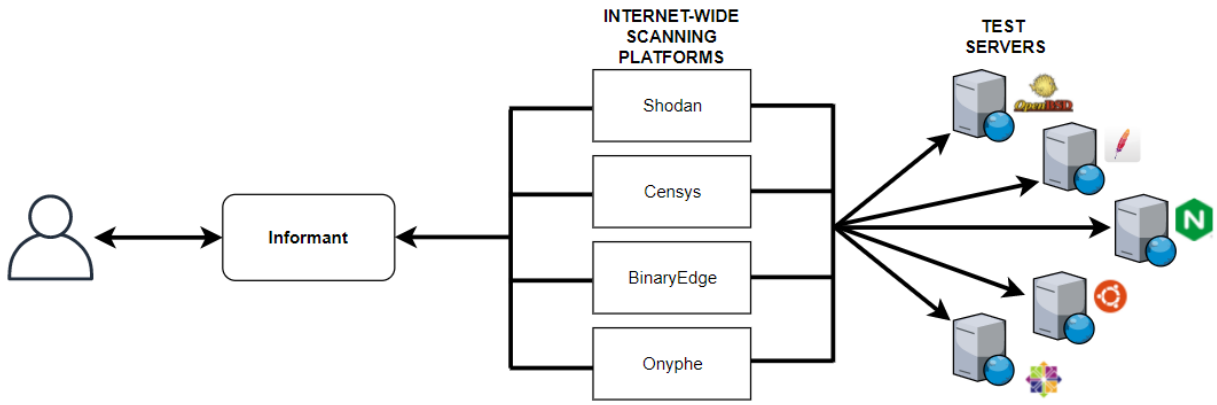


Figure 65: Data Merging Experiment Setup

Each of the five deployed cloud servers for the experiment have varying services and operating system configurations. Each was deployed 25 days before the experiments to ensure that all four IWS platforms had ample time to identify and scan each asset on all protocols. The 25 day timespan was selected based on research by Li *et al.* (2021), who found Shodan to have a delay of up to 24 days when scanning for certain TCP ports. The additional day was added to mitigate potential time zone issues and act as a buffer for any discrepancies in scanning frequencies. The configuration of each test server can be observed in Table 12.

Table 12: Test Server Configurations

Server ID	Country/Location	Operating System	Services	Exposed Ports
1	UK - London	Windows 7	Microsoft RDP	3389
			Internet Information Services 10	80
2	UK - London	Linux - Ubuntu 18.04	Apache 2.4.46	80
			Apache 2.4.46	443
			OpenSSH 7.6p1	22
3	UK - London	Linux – Centos 6	Nginx 1.19.8	8443
			vsFTP 3.0.3	21
4	UK - London	Linux – Ubuntu 20.04	Nginx 1.19.8	80
			OpenSSH 8.2	22
5	UK - London	Linux – Centos 8	OpenSSH 7.8p1	22
			Elasticsearch REST API 6.8.1	9200
			Postfix 2.10.1	25
			Nginx 1.18.0	8080

3.4.3 Experiment 2 – Common Platform Enumeration Construction

A core component within Informant is the ability to construct CPEs accurately based on service banner attributes gathered from IWS platforms, a complex and challenging process to perfect, as can be observed from the implementation stage and related past research (Genge and Enăchescu, 2016; Sanguino and Uetz, 2017; O'Hare, Macfarlane and Lo, 2019). Without this process, the known-vulnerability assignment feature would not be achievable.

This feature will be assessed via manual and comparative assessment methods. The manual assessment process will look to evaluate the accuracy of the CPE construction algorithm implemented, with methods from previous research mirrored to achieve this. Generated CPEs will be compared against CPEs constructed manually with precision and recall factors used for quality evaluation (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019). The results from the experiment will mathematically quantify the overall success or failure of this aspect of the project. The comparative assessment will evaluate the CPE construction accuracy of Informant against similar industry tooling performing the same task. Unfortunately, as gaining access to many of these tools is not feasible, this comparative analysis will only consider OpenVAS. The previous cluster of purpose-deployed UK-based servers will be used as a small sample set for this comparative analysis stage, permission of which was granted to perform scans.

3.4.4 Experiment 3 – Common Vulnerability Enumeration Assignment

The process for evaluating the CVE assignment process in the review literature is similar to the comparative assessment section described for experiment 2. The current approaches in literature involve comparing obtained outcomes against results produced by OpenVAS, an approach that will be replicated during this experiment. Additionally, Scout was also considered for the comparative process. However, as it employs an identical approach to CVE assignment as Informant and due to its minimal support for protocols, it was excluded from this comparison. Nessus was also considered; however, the free 'essential' edition obtained did not appear to provide the required CVE assignment functionality to compare the product fairly. Hence it was also excluded from the comparison.

Due to a lack of reproducible experiments within literature, the producers of Scout, O'Hare, Macfarlane and Lo (2019) proposed and adopted a well thought out approach to the CVE assignment evaluation process. For this, precision, recall and F1 metric measures were adopted to quantify the accuracy of the overall process. Due to the replicable nature and comprehensive series of evaluation metrics this proposed approach offers, it was adopted for this experiment.

3.4.5 Experiment 4 – Output Comparison of Similar Tooling

The final experiment will be another comparative assessment, comparing the overall capabilities of Informant against other similar tools. This experiment aims to evaluate how the developed solutions features and functionality compare against current tooling available within literature and industry. The tools which will be included within the comparison are OpenVAS, ShoVAT, Scout, and Nessus. The key features and capabilities of each will be analysed and compared.

CHAPTER 4

4 RESULTS

4.1 INTRODUCTION

The following section will provide a clear explanation and analysis of results obtained from the four experiments as set out in section 3.4 above. To ensure reliable results and that the experiments were fair, as many parameters as possible remained static. To achieve this, all data retrieved from IWS platforms during these experiments was gathered on the day of the experiment, within a two-hour time frame between 1200 and 1400 hours British summer time. Unless explicitly stated otherwise. This period was kept to a minimum to minimise the effects of differing IWS platform scanning frequencies as highlighted in past research (Li *et al.,* 2021). Helping to ensure a fair comparison of data gathered from each IWS platform could be drawn. PDNS data gathered for experiments was collected in a similar manner, on the same day of each experiment, within a slightly larger five-hour time frame between 1200 and 1700 hours. The larger time frame was necessary due to the sheer quantity of data and sophisticated rate-limiting features in place on free PDNS sources. Furthermore, another fundamental data element of this project, the NVD remained constant throughout each experiment. For this, an up to date copy of the NVD obtained on 30 March 2021 was used.

4.2 EXAMPLE OPERATION

Upon completion and integration of all implemented components, Informant was put to the test in a real-world scenario to ensure all aspects of the tool meet the project objectives. This example operation was fundamental to ensuring all the tools combined features work and were displayed as intended. A mediocre sized - /23 block - subsection of the Abertay University network was selected as the designated target for this example operation. A summary of key results from this operation can be found in Table 13.

Table 13: Example Operation Result Summary

Total Identified Assets	Number of Secure Assets	Number of Risky Assets	Number of External DNS Records	Number of Assets with Default Config Vulnerabilities	Number of Assets Susceptible to rDDoS
90	67	23	5	14	2

As observed in Table 13, a total of 90 assets were identified, 67 of which were classified as secure whilst the remaining 23 were flagged as potentially risky. Further analysis of the data retrieved from IWS platforms uncovered that of the four IWS platforms, BinaryEdge, in this case had the most far-reaching

scan coverage. Detecting all 90 IP addresses, a breakdown of the results is shown in Figure 66. This result clearly illustrates the discovery benefit gained from using multiple IWS platforms—vastly improving on the capabilities of past CAR projects found in literature that rely solely on one IWS data source (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019).

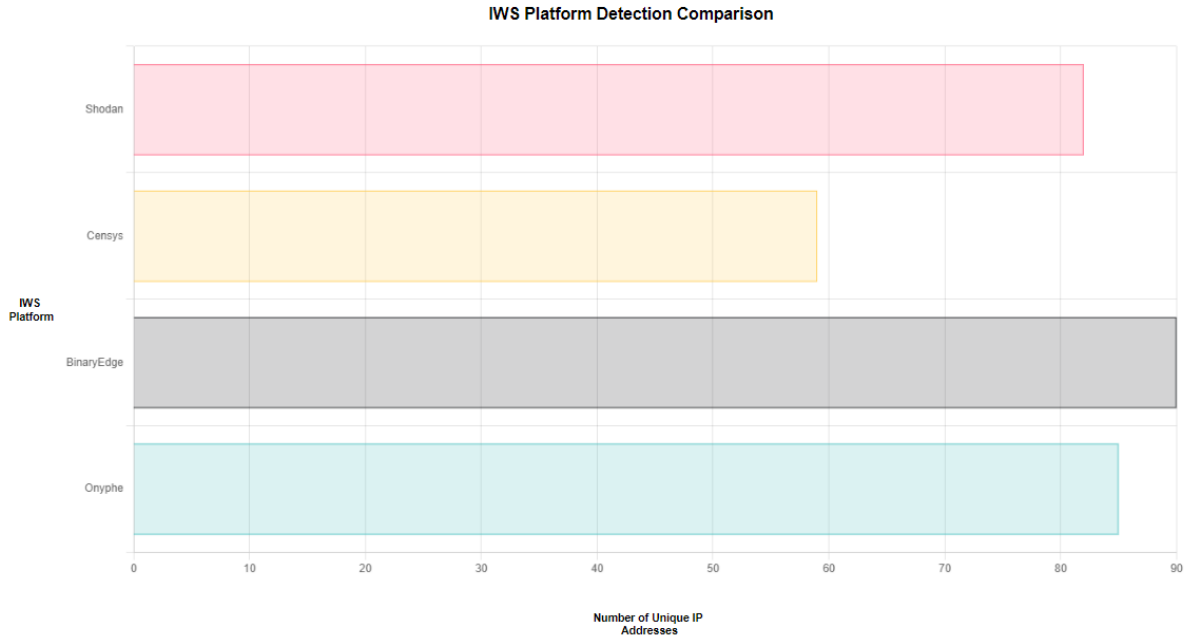


Figure 66: Example Operation - IWS Platform Comparison

The merging of banner information also resulted in vastly greater data collection, as shown in Figure 67, where several attributes are combined to form a complete picture of an individual asset. In this case, BinaryEdge was the only platform to detect and enumerate service banner information on port 161, and Censys was the only platform to retrieve OS data accurately. Censys was also found to contain the most recently obtained data on port 22. Thus, the corresponding service banner data from BinaryEdge on port 161 and Censys on port 22 was used to form the merged banner for this asset.

Timestamp	IP	ASN	Domains	Organisation	Country	OS	Ports	Banners
Thu, 01 Apr 2021 18:48:00 GMT	[redacted]	AS786		Abertay University	United Kingdom	Cisco IOS	161 22	Cisco IOS 1.25 (22) Cisco SNMP service (161)

Source	ASN	IP Address	Domains	Organisation	Country	OS	Ports	Services
_shodan	AS786	[redacted]		Abertay University	United Kingdom		22	
_censys	786	[redacted]			United Kingdom	Cisco IOS	22	Cisco IOS 1.25 (22)
_binaryedge		[redacted]					22,161	Cisco SSH 1.25 (22) Cisco SNMP service (161)
_onyphe	AS786	[redacted]		Jisc Services Limited	GB	Unknown,IOS	22	SSH 1.25 (22)

Figure 67: Example Operation - Individual Asset IWS Data Merge

Using the automated data visualisations integrated into Informant, the overall network port exposure can be visualised immediately, allowing for unusual ports to be identified quickly. The port exposure from this example operation can be seen in Figure 68. Using this data, insecure and potentially forgotten assets and services running on unusual ports can easily be spotted for manual investigation.

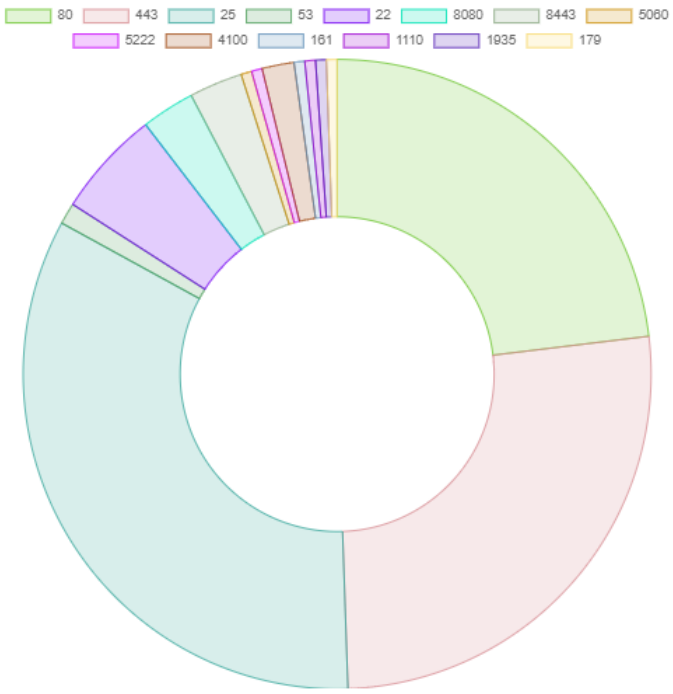


Figure 68: Example Operation - Overall Port Exposure

Analysing the results from the vulnerability capabilities of Informant, the overall web design was discovered to be clear and intuitive as desired. The main project overview page highlighted and prioritised current threats and issues within the selected time frame. A further drilldown of each asset can be obtained by clicking on the IP address associated with each asset, allowing for a further in-depth overview of all information available on a given asset. An example of this can be found in Figure 69.

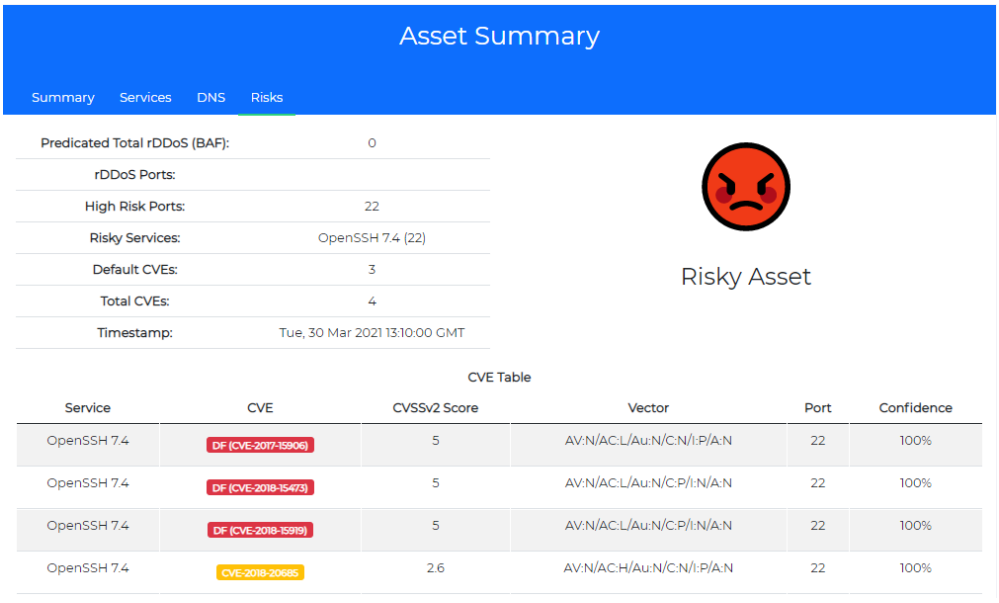


Figure 69: Individual Asset Overview

An overview showing which ports are associated with vulnerable services – see Figure 70 - helps depict which services are vulnerable across the targeted online estate, allowing for quick and efficient detection

of affected assets. This gives security analysts and network administrators the ability to quickly identify groups of assets affected by known-vulnerabilities. The ingested PDNS data detected 20 TLDs and 578 unique subdomains from 989 records that were all enriched with ASN and netname data to help locate external assets, out with the data retrieved from IWS platforms within the original CIDR range. The PDNS feature can also be used to highlight DNS records that are potentially dangling or stale, a relatively common occurrence within large organisations with poor DNS hygiene. In this case, 5 PDNS were flagged as being external assets, as shown in Figure 71.

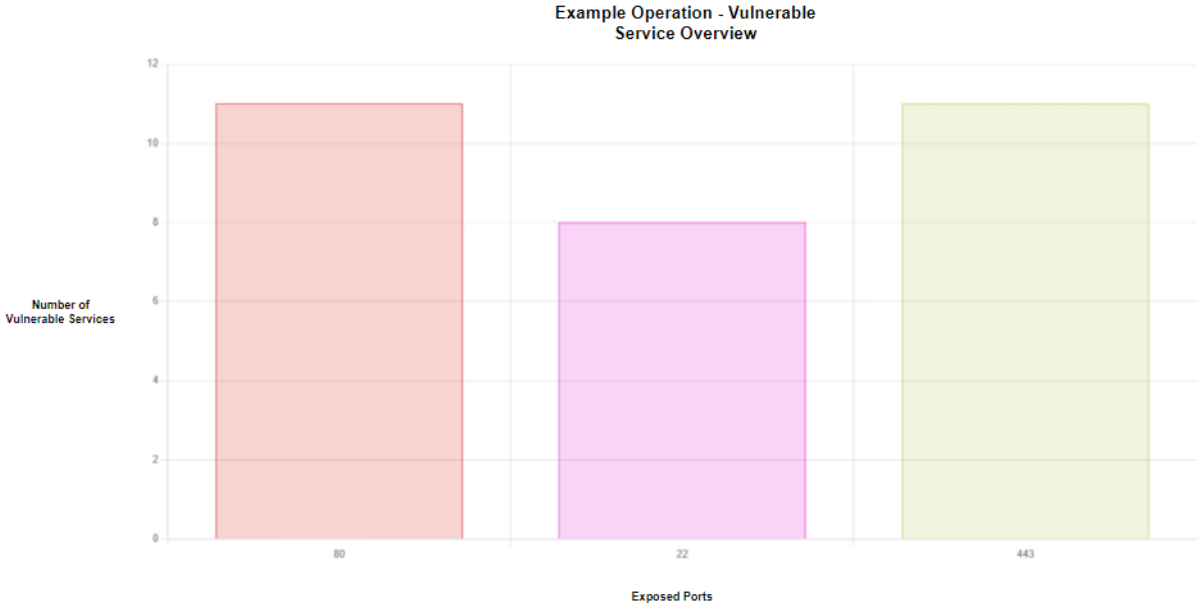
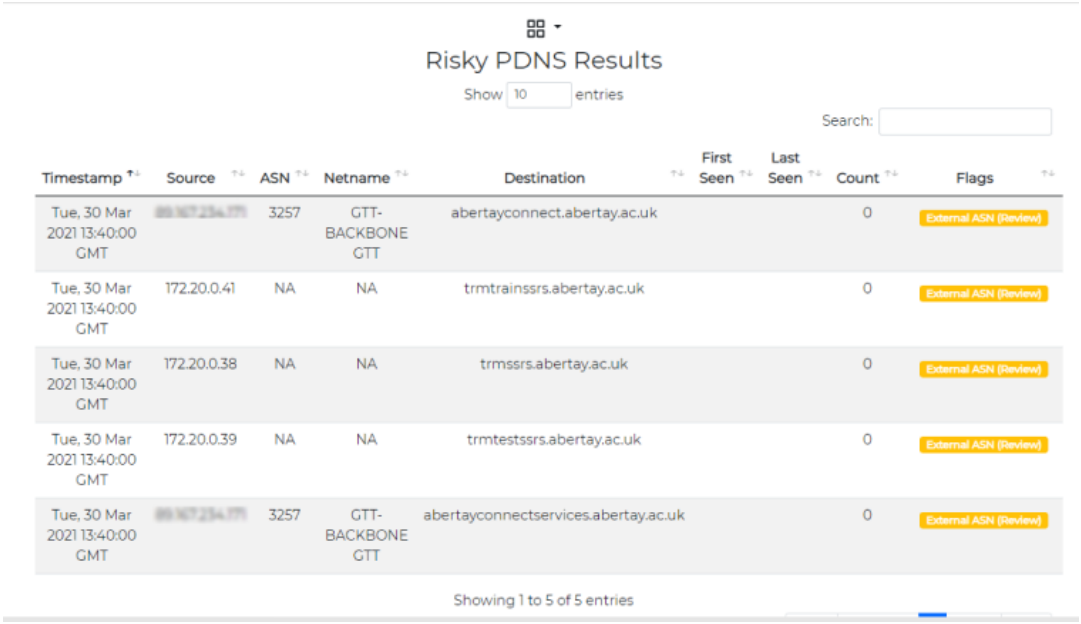


Figure 70: Example Operation - Vulnerable Services Sorted by Ports



Timestamp	Source	ASN	Netname	Destination	First Seen	Last Seen	Count	Flags
Tue, 30 Mar 2021 13:40:00 GMT	[redacted]	3257	GTT-BACKBONE GTT	abertayconnect.abertay.ac.uk			0	External ASN (Review)
Tue, 30 Mar 2021 13:40:00 GMT	172.20.0.41	NA	NA	trmtrainssrs.abertay.ac.uk			0	External ASN (Review)
Tue, 30 Mar 2021 13:40:00 GMT	172.20.0.38	NA	NA	trmssrs.abertay.ac.uk			0	External ASN (Review)
Tue, 30 Mar 2021 13:40:00 GMT	172.20.0.39	NA	NA	trmtestssrs.abertay.ac.uk			0	External ASN (Review)
Tue, 30 Mar 2021 13:40:00 GMT	[redacted]	3257	GTT-BACKBONE GTT	abertayconnectservices.abertay.ac.uk			0	External ASN (Review)

Showing 1 to 5 of 5 entries

Figure 71: PDNS External DNS Records

Reviewing the results of Figure 71, the external PDNS detection method employed proved to provide unexpected results. Three of the five DNS records flagged as external can be seen to be non-public IP addresses that are presumably for internal use only. These addresses, which reside within the '172.20.0.0' range, are of particular interest as they were unexpected and theoretically should not show up in PDNS data due to the nature of how it is collected. Such addresses give a potential insight into the internal network infrastructure of the university's intranet. Investigating further into the DNS data surrounding the private IP addresses, they were found to all be obtained by the forward DNS integration of DNSGrep and are thought to have been detected by PDNS due to misconfigured DNS records. This theory is further backed by the naming convention of each associated subdomain associated with the private IP addresses exposed, which appear to have been created during training and testing exercises. The remaining two flagged records were found to be correctly identified as having the potential of being dangling DNS records, with both pointing towards a third-party platform aimed at supporting student career development. In this case, manual analysis of both flagged records indicated that both DNS records were still in use and not dangling at this time. However, this feature successfully identified associated external assets that IWS platforms cannot attribute to the targeted university, as they are out with the original network ASN.

The final aspect of reviewing the core capabilities of Informant was to ensure that an interactive exposed attack surface network map, using both IWS and PDNS data, could be automatically generated. Such a diagram can be seen in Figure 72 and is incredibly useful for deriving the use case of assets via analysis of associated domains. As shown in the below exposed attack surface network map, the correlated data can help identify critical assets, such as mailing, backup and archive servers. Furthermore, in many cases, assets without associated PDNS records are often of most interest, as a lack of DNS data can often be attributed to assets that are not intended to be exposed to the internet. In this specific case, a notable asset of interest would be an exposed Vivotek camera system located behind a single factor HTTP authentication portal. The exposed asset has no associated DNS records and as such is likely to be the cause of an accidental device exposure.

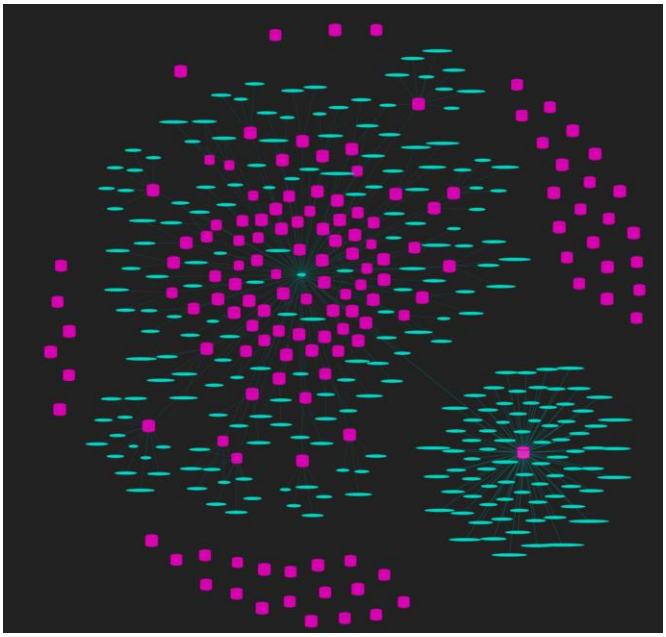


Figure 72: Abertay Exposed Attack Surface

To evaluate the tool at scale, four /16 network blocks of UK-based universities were analysed using Informant. In total, 1147 exposed ports and 953 services were processed, with a staggering 50% of detected assets found to be vulnerable. Comparing this with similar tooling within literature, the authors of ShoVAT, Gene & Enăchescu (2016) found that of the educational institutions they assessed, most CVSS scores on average were around 5, with a spread across the entire CVSS range of 0-10. In a comparable experiment, the authors of Scout, O'Hare, Macfarlane and Lo (2019) obtained similar results, with an average CVSS score of around 5. However, of the educational institutions tested by ShoVAT and Scout, both were found to have an outlier, with one educational institution in each having a noticeably smaller range of CVSS scores. Reproducing this experiment for Informant, a similar conclusion can be drawn as depicted in Figure 73. A consistent average CVSS score of around five can be found in each result, with one of the four universities having a noticeably smaller overall CVSS score range. However, a common spread across the 0-10 CVSS score spectrum can be seen to remain consistent.

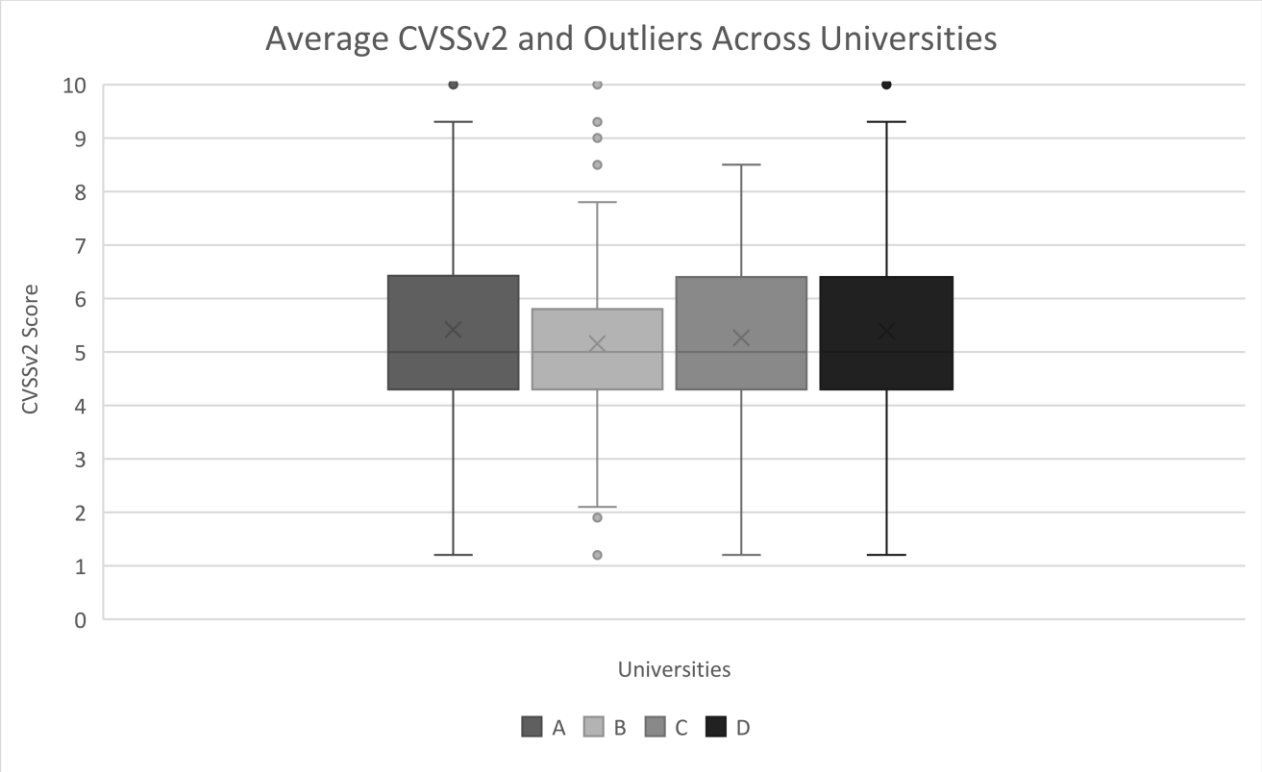


Figure 73: Average Identified CVSSv2 Across Universities

Of the 5515 CVEs detected by Informant across the four universities, 48% were flagged as being vulnerable on their default service configuration. In Figure 74, a comparison of CVEs flagged as being vulnerable by default against non-default CVEs shows the spread of CVEs vulnerable by default across the universities. The raw data produced by Informant can be used to create the graphs seen in this section, in addition to the automatically generated graphs. Informants ability to extract raw data directly from the web interface and automatically generate useful graphics allows for easy report generation, putting the tool in a league of its own compared to past developed CAR tools (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019).

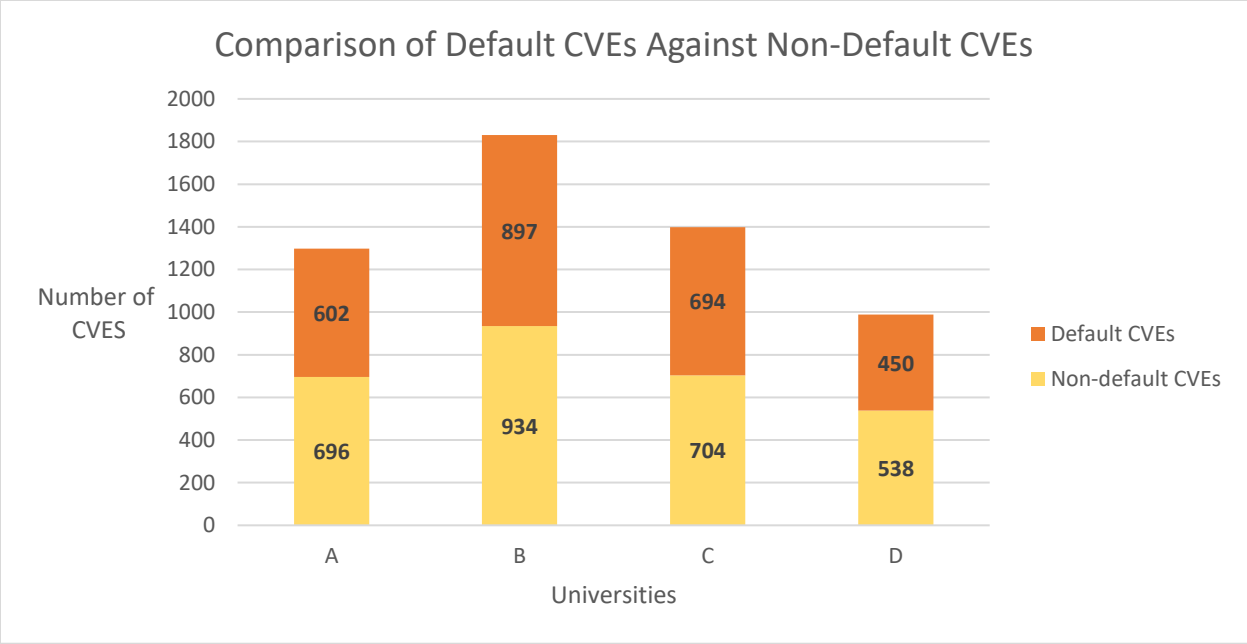


Figure 74: Amount of Non-default and Default CVEs per University

4.3 Experiment 1 – Internet-wide Scanning Data Merging

This experiment aimed to analyse how effective the implemented algorithm for merging data retrieved from IWS platforms was. To achieve this, a non-bias set of five servers were configured and deployed according to the experiment plan in section 3.4.2 and configuration setup in Table 12. The results produced from the automated merging process are then compared against the actual service configuration of the deployed servers to determine the overall accuracy of the approach. Table 14 shows this comparisons, highlighting which platform successfully detected key individual IWS platform attributes, whereby '✔' denotes an accurate match, '✘' denotes an inaccurate or no match and '—' denotes that Informant does not consider the data from the selected platform. As was seen during the limited singular unit testing of the merging procedure in Figure 46, the function was found to perform well. Hence, it is expected that on this more diverse range of services the feature will continue to provide more accurate results than solely relying on one platform.

Table 14: Experiment 1 - Key Attribute Merging Overview

Key Attribute Merging Comparison					
	Server ID	Country	Operating System	Services	Exposed Ports
Shodan	1	✓	–	✓	✓
	2	✓	–	✓	✓
	3	✓	–	✓	✓
	4	✓	–	✓	✓
	5	✓	–	✗	✗
Censys	1	✓	✗	✓	✓
	2	✓	✓	✓	✓
	3	✓	✓	✗	✗
	4	✓	✓	✓	✓
	5	✓	✗	✗	✗
BinaryEdge	1	–	–	✓	✓
	2	–	–	✓	✓
	3	–	–	✓	✓
	4	–	–	✓	✓
	5	–	–	✓	✓
Onyphe	1	✓	✓	✗	✓
	2	✓	✓	✗	✗
	3	✓	✓	✓	✓
	4	✓	✓	✗	✓
	5	✓	✓	✗	✗

Table 14 shows the results from the process of cross-referencing the known service configuration of each test server against the key attributes retrieved from each IWS platform by Informant. As can be seen above, Informant clearly benefits from data provided by multiple platforms in all test scenarios. For example, in the case of server 5, only BinaryEdge was capable of accurately detecting the exposed ports and service banners fully. A full breakdown indicating which platform was selected for each attribute by the merging process can be seen in Table 15, which clearly shows what source makes up each aspect of a fully merged record. Interestingly, of the four IWS platforms, BinaryEdge was found to be capable of successfully gathering the most information on assets. However, due to Informant's previously discussed design limitations – related primarily to credit limitations - not all data available on BinaryEdge is considered by Informant, limiting its full data retrieval potential within Informant.

Table 15: Experiment 1 - Merged Record Attribute Sources

Server ID	ASN	Country	Company	Hostnames	Domains	Ports	Protocols	OS	Banners
1	Shodan	Censys	Shodan	Shodan Onyphe	Shodan Onyphe	Shodan	Censys	Onyphe	Shodan Censys Onyphe
2	Shodan	Shodan	Shodan	Shodan Onyphe	Shodan Onyphe	Shodan	Censys	Censys	Shodan BinaryEdge Onyphe
3	Shodan	Shodan	Shodan	Shodan Onyphe	Shodan Onyphe	Shodan	Censys	Censys	Censys BinaryEdge
4	Shodan	Shodan	Shodan	Shodan Onyphe	Shodan Onyphe	Shodan	Censys	Censys	Censys
5	Shodan	Shodan	Shodan	Shodan Onyphe	Shodan Onyphe	Shodan BinaryEdge	Censys	Onyphe	Shodan BinaryEdge

Analysing the final five merged records found that an accurate record was formed in all five cases using a combination of data. Thus, helping to mitigate issues highlighted in the literature concerning obtaining enough data to conduct CPE banner construction (O'Hare, Macfarlane and Lo, 2019). However, other

factors such as the timelessness of platform scan frequencies can still adversely affect the accuracy of results, although this issue is clearly minimised due to the use of multiple sources. From Table 15, each of the IWS platforms appear to be equally competitive regarding the frequency of scans, as illustrated by the even attribute selection spread across the banners portion of merging. Overall, the experiment proves that the adopted merging algorithm successfully improves the accuracy and relevance of IWS data obtained. All test servers benefited from using data from multiple IWS platforms in this test.

4.4 Experiment 2 – Common Platform Enumeration Construction

Using a combination of the results from the example operation and experiment 1, a total of 49 unique services were identified. For this experiment, the Informant tool used the banner data from within the merged records of these results to construct CPE values that could then be compared against manually produced CPE values known to be correct. The decision to combine the example operation and the experiment 1 dataset was made to ensure an unbiased and realistic sample of banner data was gathered on a wide spread of services across various ports, keeping the experiment as fair as possible. This allowed for the performance of the Informant CPE construction algorithm to be measured accurately and for this experiment to be reproducible. A summary of the overall results from the manual assessment of this feature can be found in Table 16.

Table 16: Overview of CPE Construction Results

Total Unique Identified Services	Number of Positive Results	Number of False Results	Number of False Positives	Number of False Negatives	Number of Unique Obfuscated Services
29	22	7	1	2	20

Table 17: CPE Construction Confusion Matrix

	Actual: Match	**Actual:** No Match
Predicted: Match	21	1
Predicted: No Match	2	8

Of the 49 unique services detected from the two combined datasets, 20 were classified as obfuscated due to a lack of retrievable version information. These services primarily consisted of web server services such as Nginx and Apache. In the confusion matrix within Table 17, the true positive results represent the correct CPEs. The true negative results show the shortcomings of the NVD as the majority of these results were related to CPEs not existing within the NVD dataset. Although reasons for CPEs not occurring in the

NVD vary vastly; in this instance, it was found to be due to overlapping product life cycles and new product versions not having a corresponding CPE within the database. Simply manually adding the missing CPE values to the database proved to be of no use due to the CVE dataset also not containing the correct information relating to the service.

Analysing the results in Table 16, it can be derived that only one result was assigned an incorrect CPE value and was associated with another product, resulting in a false positive. Examining the reason behind this, found that Informant produced an incorrect CPE of 'cpe:2.3:a:cisco:ios:*:*:*:*:*:*:*:*' from a service banner of 'Cisco IOS 1.25'. Whereby, the appropriate correct CPE value of, 'cpe:2.3:o:cisco:ios:12.5\(22\)md:*:*:*:*:*:*:*' should have been produced. However, this CPE is an operating system component. A factor that Informant does not consider when constructing CPE values. Identifying a flaw within the developed approach, as the assumption was made that only a application part value of 'a' would need to be considered when constructing CPEs.

The two false negative results were found to be caused by incorrectly identified services. The first was related to the provided service banner of 'Tandberg-4144 VoIP server X12.7.1', which could not be verified as the true CPE differs drastically from this naming convention and is represented by the CPE 'cpe:2.3:a:cisco:telepresence_video_communication_server:x12.7.1:*:*:*:*:*:*:*'. This CPE has little similarity with the inputted service banner data, and therefore, no CPE could be verified by Informant, as seen in Figure 75. This identifies an issue with Informants reliance on the Levenshtein algorithm for CPE validation. As if garbage data is inputted, garbage data will be returned, and in this specific case, no data is returned - due to the set threshold in place to prevent excessive false positives. Removing or lowering this threshold would only yield a higher false positive rate.

```
Original service: Tandberg-4144 VoIP server X12.7.1
Original CPE: cpe:2.3:a::tandberg-4144 voip server:x12.7.1:*:*:*:*:*:*:*
[INFORMANT] Verifying CPE Value
Generated CPE (after exceptions): cpe:2.3:a::tandberg-4144 voip server:x12.7.1:*:*:*:*:*:*:*

Generated CPE could not be verified. Using generated CPE value.
```

Figure 75: Informant CPE Construction Output for 'Tandberg-4144 VoIP server X12.7.1'

The same issue caused the second false negative, which was related to receiving 'mod_sftp 0.9.9' as an input, causing Informant to produce 'cpe:2.3:a::mod_sftp:0.9.9:*:*:*:*:*:*:*' when the actual CPE of 'cpe:2.3:a:proftpd:proftpd:1.3.3:rc1:*:*:*:*:*:*' is significantly different – vendor, product and version is completely different. Informants attempt to produce this CPE can be seen in Figure 76. Although Informants method of ingesting individual attributes to produce an initial CPE is reliable, future improvement addressing the tools reliance on the Levenshtein algorithm should be considered.

```
Original service: mod_sftp 0.9.9
Original CPE: cpe:2.3:a::mod_sftp:0.9.9:*:*:*:*:*:*:*
[INFORMANT] Verifying CPE Value
Generated CPE (after exceptions): cpe:2.3:a::mod_sftp:0.9.9:*:*:*:*:*:*:*

Generated CPE could not be verified. Using generated CPE value.
```

Figure 76: Informant CPE Construction Output for 'mod_sftp 0.9.9'

The results in Table 17 were scientifically quantified using the precision, recall and F-measure as declared in the experimentation section 3.4.3. Such methods are often used when quantifying web vulnerability scanning tools; however, the authors O'Hare, Macfarlane and Lo (2019) demonstrated how well these measurement methods are suited towards CPE construction. For the proceeding calculations, the confusion matrix in Table 17 and statistics in Table 16 were continuously referred to.

The precision is the ratio of correctly matched CPEs against the total number of all CPEs matched. This is used to quantify the accuracy of the approach, with the equation and result denoted in Equation 2.

Equation 2: Informant's CPE Construction Precision

$$Precision = \frac{TP}{TP + FP} = \frac{21}{21 + 1} = \frac{21}{22} = 0.95$$

The recall mathematically determines what ratio of correct CPEs were actually matched correctly. This is done by dividing the ratio of correctly matched CPEs against the potentially correct matches. Equation 3 depicts this process mathematically.

Equation 3: Informant's CPE Construction Recall

$$Recall = \frac{TP}{TP + FN} = \frac{21}{21 + 7} = \frac{21}{28} = 0.75$$

The F-measure (F_1 score), is a measurement of effectiveness. Utilising both the precision and recall values to compute the harmonic mean. Equation 4 shows this calculation.

Equation 4: Informant's CPE Construction F_1 Score

$$F_1 = \frac{2x\ Precision + Recall}{Precision + Recall} = \frac{2\ x\ 0.95\ x\ 0.75}{0.95 + 0.75} = \frac{1.43}{1.70} = 0.84$$

The above metrics are directly comparable to the CPE construction process of Scout due to the overlapping evaluation methodology employed. Although, it must be considered that the evaluation of Scout was restricted to a finite amount of ports and services, which potentially could work in Scouts favour. Due to the more sophisticated nature of Informant, no such restrictions on ports nor services are in place. Additionally, a direct comparison to ShoVAT using these metrics was not possible due to a lack of detail and required figures surrounding experiments conducted. A comparison table comparing Scout and Informant using the above metrics can be found in Table 18.

Table 18: Comparison of CPE Construction Evaluation Metrics for Informant and Scout

Informant CPE Construction	
Precision	0.95
Recall	0.75
F_1 Score	0.84
Scout CPE Construction	
Precision	0.94
Recall	0.56
F_1 Score	0.68

The results in Table 18 shows that the CPE construction approach of Informant outperforms that of Scout. With Informant having a more accurate and correct CPE construction rate which is indicated by the higher precision and recall values. The F1 score indicates that the overall accuracy of Informants CPE construction approach slightly outperforms Scout. This result can be associated with the additional ratio and manual service exception features implemented into Informant.

A comparison of Informant against ShoVAT, which is more similar in capabilities than Scout due to its unrestrictive ability to construct banners across several protocols, can be conducted via more traditional mathematical analysis, as observed in Table 19. These results indicate that Informant is less likely to return both a false positive and false negative result. The result that stands out the most is Informants better precision, with a lesser chance than ShoVAT of producing a false positive result. Often an impactful decisive factor when selecting a security tool.

Table 19: ShoVAT and Informant CPE Construction Result Comparison

Tool	Unique Banners	False Results	Number of False Positives	False Positives %	Number of False Negatives	False Negatives %
ShoVAT	296	46	23	7.77%	23	7.77%
Informant	29	7	1	3.45%	2	6.90%

Comparatively comparing Informants CPE construction approach against the freely available industry recognised OpenVAS tool, found that during testing, Informant could correctly detect two more CPE banners than OpenVAS. The results from this experiment are presented in Table 20. They show that both tools can only produce an accurate CPE when enough data is available, with an emphasis on the importance of version data. For this experiment, additional variants of services were configured across the five test servers to further test and compare the CPE construction capabilities of the two tools. One banner was falsely advertised via the service banner 'HTTP Server 2.4.46', resulting in Informant throwing a false positive of 'apache:http_server:2.4.46'. OpenVAS was found not to provide a service banner for this service, indicating that active scanning measures detected the falsified service banner, meaning that CPE construction within OpenVAS does not solely rely on banner data. However, OpenVAS failed to correctly enumerate two OpenSSH banners containing additional OS meta-data, in both cases returning CPEs without the correct update component. Whilst Informant successfully identified these services. From these results, it can be concluded that Informant can handle a more diverse range of advertised service

banners but is unable to adopt the same calibre of verification methods that active scanning allows for. Potentially leading to a higher rate of false positives.

Table 20: Comparison of Informant CPE Construction Results Against OpenVAS

Service	Displayed Service Banner	Official CPE	OpenVAS	Informant
Microsoft IIS 10.0	'Microsoft IIS httpd 10.0'	'microsoft:internet_information_server:10.0'	✓	✓
Microsoft IIS 10.0	'IIS 10.0'	'microsoft:internet_information_server:10.0'	✓	✓
Apache 2.4.46	'Apache httpd 2.4.46'	'apache:http_server:2.4.46'	✓	✓
Apache 2.4.46	'HTTP Server 2.4.46'	'apache:http_server:2.4.46'	✓	✓
Apache 2.4.7	'HTTP Server 2.4.46'	'apache:http_server:2.4.7'	X	X
Apache 2.4.6	'Apache httpd'	'apache:http_server:2.4.6'	X	X
OpenSSH 7.6p1	'OpenSSH 7.6p1'	'openbsd:openssh:7.6:p1'	✓	✓
OpenSSH 7.6p1	'OpenSSH 7.6p1 Ubuntu-4ubuntu0.3'	'openbsd:openssh:7.6:p1'	X	✓
Nginx 1.19.8	'nginx 1.19.8'	'nginx:nginx:1.19.8'	✓	✓
Vsftpd 3.0.3	'vsFTPd 3.0.3'	'beasts:vsftpd:3.0.3'	X	X
OpenSSH 8.2	'OpenSSH 8.2'	'openbsd:openssh:8.2'	✓	✓
OpenSSH 8.2p1	'OpenSSH 8.2p1'	'openbsd:openssh:8.2:p1'	✓	✓
OpenSSH 8.2p1	'OpenSSH 8.2p1 Ubuntu-4ubuntu0.1'	'openbsd:openssh:8.2:p1'	X	✓
Elasticsearch 6.8.1	'Elasticsearch REST API 6.8.1'	'elasticsearch:elasticsearch:6.8.1'	X	X
Postfix 2.10.1	'postfix smtpd 2.10.1'	'postfix:postfix:2.10.1'	✓	✓
Nginx 1.18.0	'nginx 1.18.0'	'nginx:nginx:1.18.0'	✓	✓
Nginx 1.18.0	'nginx'	'nginx:nginx:1.18.0'	X	X

4.5 Experiment 3 – Common Vulnerability Enumeration Assignment

For this experiment, two variants of Nginx and OpenSSH were investigated to evaluate and compare the CVE assignment capabilities of Informant against the total quantity of available CVEs within the NVD. Servers 4 and 5 used in the previous experiments were selected for this experiment and reconfigured

accordingly, with Nginx and OpenSSH. The produced results can be viewed in Table 21. Scout was not included in this comparison as the approach used for CVE assignment by Informant is the same, with the only difference being the more recent local NVD database. Hence, the results would not differ from that of Informant.

Table 21: CVE Assignment Comparative Results

Service	NVD	OpenVAS	Informant
Nginx 1.15.0	7	5	3
Nginx 1.14.0	8	6	4
OpenSSH 7.6p1	9	0	9
OpenSSH 7.3	18	12	16

The results in Table 21 show that the NVD outperforms both Informant and OpenVAS in all four scenarios, with Informant beating OpenVAS in two of the four cases. An unexpected result considering the vulnerability assignment feature of Informant relies on the NVD correlation between CPEs and CVEs to perform known-vulnerability assessment. Theoretically meaning that Informant and the NVD results should match. However, it is speculated that underperforming results from Informant must be related to issues with the locally obtained NVD database and previously drawn upon issues surrounding the NVD and missing CPEs (Khadilkar, Rachapalli, and Thuraisingham, 2010).

To begin analysing the results of this experiment, the Nginx 1.14.0 service results were broken down by outputted CVE and compared by source. The breakdown can be observed in Table 22. From these results, Informant was found to exclude four vulnerabilities within the NVD, whereas OpenVAS was found to exclude only two. The primary hypothesised reason for the inconsistency in results by Informant was thought to be down to poor data correlation as already mentioned. To find out if this was the cause, the presence of the four CVEs missed by Informant was successfully located within the local copy of the NVD, with both the 'vulnerable_product' and 'vulnerable_configuration' fields analysed and cross-referenced with the official NVD online database. The findings from the cross-referencing revealed that the local copy of the NVD obtained via the cve-search utility had inconsistencies within the CPE to CVE data correlation. Thus, highlighting an issue within a key dependency of Informant.

Table 22: Nginx 1.14.0 CVE Assignment Report

NVD	OpenVAS	Informant
CVE-2009-4487		
CVE-2018-16843	CVE-2018-16843	CVE-2018-16843
CVE-2018-16844	CVE-2018-16844	CVE-2018-16844
CVE-2018-16845	CVE-2018-16845	CVE-2018-16845
CVE-2019-9511	CVE-2019-9511	
CVE-2019-9513	CVE-2019-9513	
CVE-2019-9516	CVE-2019-9516	
CVE-2019-20372		CVE-2019-20372
Differences	+0/-2	+0/-4
	Differences	+1/-3

To scientifically quantify the results, the precision can be calculated to show the percentage of correct vulnerabilities against the total number of vulnerabilities within the NVD. This was done by firstly working

out the true and false positive values. The true positives represent the results detected by both Informant and OpenVAS. The false positives only relate to Informant results. The precision calculation for this service can be seen in Equation 5.

Equation 5: Informant CVE Assignment Precision Against OpenVAS (Nginx 1.14.0)

$$Precision = \frac{TP}{TP + FP} = \frac{3}{3 + 1} = \frac{3}{4} = 0.75$$

Informants precision of 75% for the Nginx 1.14.0 service shows how the CAR approach adopted compares to the active scanning measures used by OpenVAS to detect vulnerabilities. In addition, the recall can be calculated to quantify the proportion of correct, true vulnerabilities returned using the true positives and false negatives, with false negatives representing the different CVEs present in the OpenVAS results compared to Informant. The recall calculation can be seen in Equation 6.

Equation 6: Informant CVE Assignment Recall Against OpenVAS (Nginx 1.14.0)

$$Recall = \frac{TP}{TP + FN} = \frac{3}{3 + 3} = \frac{3}{6} = 0.50$$

Informants recall of 50% for the Nginx 1.14.0 service quantifies the absoluteness of results provided by the CVE assignment process. This means that Informant is capable of identifying 50% of known-vulnerabilities for the Nginx 1.14.0 service. The F-measure (F1 score) can be used to measure the effectiveness of the CVE assignment task. Utilising both the precision and recall values to compute their harmonic mean. Equation 7 shows this calculation.

Equation 7: Informant CVE Assignment F-Measure Score Against OpenVAS (Nginx 1.14.0)

$$F_1 = \frac{2x\ Precision + Recall}{Precision + Recall} = \frac{2\ x\ 0.75\ x\ 0.50}{0.75 + 0.50} = \frac{0.75}{1.25} = 0.60$$

The breakdown process of Nginx 1.15.0 was repeated following the same procedure as above. The breakdown can be seen in Table 23, precision calculation in Equation 8, recall calculation in Equation 9 and F1 score in Equation 10. Further breakdowns of OpenSSH 7.6p1 and OpenSSH 7.3 were not conducted due to the inability of OpenVAS to associate CVEs with OpenSSH 7.6p1.

Table 23: Nginx 1.15.0 CVE Assignment Report

NVD	**OpenVAS**	**Informant**
CVE-2009-4487		
	CVE-2018-16843	
CVE-2018-16844	CVE-2018-16844	CVE-2018-16844
CVE-2018-16845	CVE-2018-16845	CVE-2018-16845
CVE-2019-9511	CVE-2019-9511	
CVE-2019-9513	CVE-2019-9513	
CVE-2019-9516	CVE-2019-9516	
CVE-2019-20372		CVE-2019-20372
Differences	+1/-2	+0/-4
	Differences	+1/-4

Equation 8: Informant CVE Assignment Precision Against OpenVAS (Nginx 1.15.0)

$$Precision = \frac{TP}{TP + FP} = \frac{3}{3 + 1} = \frac{3}{4} = 0.75$$

Equation 9: Informant CVE Assignment Recall Against OpenVAS (Nginx 1.15.0)

$$Recall = \frac{TP}{TP + FN} = \frac{3}{3 + 3} = \frac{3}{6} = 0.50$$

Equation 10: Informant CVE Assignment Recall Against OpenVAS (Nginx 1.15.0)

$$F_1 = \frac{2x\ Precision + Recall}{Precision + Recall} = \frac{2\ x\ 0.75\ x\ 0.50}{0.75 + 0.50} = \frac{0.75}{1.25} = 0.60$$

As shown above, the precision, recall, and F1 score are consistent against both versions of Nginx tested, primarily due to the overlap of vulnerabilities affecting both versions. Informant was found to detect the same false positive 'CVE-2019-20372' across both instances due to limitations of the CAR process, whereas OpenVAS did not associate it. By manually investigating this CVE, it is apparent that it only applies to instances where a load balancer is fronting Nginx. Therefore, it can be presumed that OpenVAS is utilising its active scanning capabilities to determine if a load balancer is in place. This highlights a fundamental weakness in the CAR process, as no additional validation is possible without interfacing with the target, which is out with the scope of this tool.

A unique feature of Informant aims to identify CVEs known to be affected by the default configuration of services. To evaluate the performance of this feature, results from the above experiments were manually analysed for CVEs that are vulnerable by default. These were compared against the results produced by Informant. As the evaluation method used in this stage does not include another tool for comparison, both Nginx and OpenSSH services can be factored into this experiment. The detected CVEs in Table 24 and Table 25 highlighted in red are vulnerable to the associated services on their base default configuration. The NVD data has been manually analysed and flagged appropriately. For this manual classification, CVEs as classified as vulnerable by default if they can be exploited on the default configuration of the service without outside contributing factors, such as dependencies, config changes or man-in-the-middle attacks. Additionally, the primary attack vector must be over the network.

Table 24: OpenSSH Default Vulnerability Breakdown

OpenSSH 7.3	
NVD	**Informant**
CVE-2007-2768	
CVE-2008-3844	
CVE-2016-8858	CVE-2016-8858
CVE-2016-10009	CVE-2016-10009
CVE-2016-10010	CVE-2016-10010
CVE-2016-10011	CVE-2016-10011
CVE-2016-10012	CVE-2016-10012
CVE-2016-10708	CVE-2016-10708
CVE-2017-15906	CVE-2017-15906
CVE-2018-15473	CVE-2018-15473
CVE-2018-15919	CVE-2018-15919
CVE-2018-20685	CVE-2018-20685
CVE-2019-6109	CVE-2019-6109
CVE-2019-6110	CVE-2019-6110
CVE-2019-6111	CVE-2019-6111
CVE-2020-14145	CVE-2020-14145
CVE-2020-15778	CVE-2020-15778
CVE-2021-28041	CVE-2021-28041
OpenSSH 7.6p1	
NVD	**Informant**
CVE-2018-15473	CVE-2018-15473
CVE-2018-15919	CVE-2018-15919
CVE-2018-20685	CVE-2018-20685
CVE-2019-6109	CVE-2019-6109
CVE-2019-6110	CVE-2019-6110
CVE-2019-6111	CVE-2019-6111
CVE-2020-14145	CVE-2020-14145
CVE-2020-15778	CVE-2020-15778
CVE-2021-28041	CVE-2021-28041

Table 25: Nginx Default Vulnerability Breakdown

Nginx 1.14.0	
NVD	**Informant**
CVE-2009-4487	
CVE-2018-16843	CVE-2018-16843
CVE-2018-16844	CVE-2018-16844
CVE-2018-16845	CVE-2018-16845
CVE-2019-9511	
CVE-2019-9513	
CVE-2019-9516	CVE-2019-20372
CVE-2019-20372	
Nginx 1.15.0	
NVD	**Informant**
CVE-2009-4487	
CVE-2018-16844	CVE-2018-16844
CVE-2018-16845	CVE-2018-16845
CVE-2019-9511	
CVE-2019-9513	
CVE-2019-9516	
CVE-2019-20372	CVE-2019-20372

From Table 25, it can be observed that Informant fails to flag the default Nginx vulnerability across both versions due to it not being allocated in the CVE assignment process. This is a direct result of inconsistencies within the local NVD and not an issue with Informant. However, in both Nginx instances, false positives are thrown. These false positives can be directly attributed to the exclusion of the previously discussed NLP summary default configuration text-mining approach from Informant. The current implemented approach only considers the 'AC' and 'AV' CVSS vector attributes, which, as speculated, can be seen not to hold enough weight or contain enough data to classify default CVEs with absolute accuracy. The hypothesis is that by implementing an additional detection mechanism such as the proposed CVE summary text-mining approach in conjunction with the implemented vector approach, there would be an exponentially lower rate of false positives. In the specific case of 'CVE-2018-16843', the proposed text-mining approach would be capable of ruling out the CVE due to the summary containing the phrase 'not compiled by default' (NIST, 2020a).

Analysing the OpenSSH results in Table 24 indicates that Informant's default CVE identifier detected one unique false negative and no false positives. These results show that the adopted approach can identify default vulnerabilities with an accuracy of 67% across the two OpenSSH versions tested. The employed technique's overall accuracy can likely be significantly improved by adding the proposed CVE summary classification mechanism. This additional proposed NLP feature has not been implemented into Informant due to time constraints, providing an exciting avenue for future work.

4.6 Experiment 4 – Output Comparison of Similar Tooling

The final experiment will be another comparative assessment, comparing the overall capabilities of Informant against other similar tools. The tools for the comparisons will include OpenVAS, Scout, and Nessus. This experiment aims to evaluate how Informants features and functionality compare against current tooling available in literature and industry. Investigating how the implemented CAR improvements stack up against other solutions. This experiment was conducted by means of a comparative feature analysis and review of each selected tool. The results of which can be found in Table 26, where '●'is used to denote a weak or no dependence, '●●'is used to denote a medium dependence and '●●● 'is used to denote a strong dependence.

Table 26: Final Tool Comparison

Tool	Passive	Active	Custom Banner	CPE/CVE	PDNS	Default Config CVE Detection	rDDoS Prediction	Historical Data
Scout	●●●	●	●●	●●●	●	●	●	●
ShoVAT	●●●	●	●●	●●●	●	●	●	●
Nessus	●	●●●	●●	●●●	●	●	●	●●
OpenVAS	●	●●●	●●	●●●	●	●	●	●●
Informant	●●●	●	●●	●●●	●●●	●●	●●	●●●

It can be seen from Table 26 that Informant offers multiple features not currently supported by current tooling. Such features include the addition of PDNS data, identification of CVEs exploitable by default and rDDoS prediction capabilities. These features all enable a more concise picture of risks within an exposed attack surface to be uncovered. The differences between the three CAR tools – Scout, ShoVAT and Informant – stand out as being the most significant, with Informant offering a vast array of additional and more refined features. In addition to the unique features of Informant, the historical data storing capability of the tool stacks up well against the traditional active based scanning solutions and is a feature no other CAR tool in literature possesses.

Although OpenVAS and Nessus are both highly capable and feature-rich tools, they lack the stealth capabilities that CAR tools such as Informant can offer. Furthermore, in contrast to Informants automated approaches, the alternative method of manually conducting and gathering data, is a time intense and tedious process. Features such as calculating the potential rDDoS throughput of an asset and gathering related PDNS data are not automated to the same extent as Informant on any available tool found in literature. Additionally, although not an evaluated metric, the speed of Informant far surpassed that of OpenVAS and Nessus.

CHAPTER 5

5 DISCUSSION

5.1 INTRODUCTION

This chapter will discuss and evaluate the reviewed literate and results from the experiments conducted in line with the overall project aims, in order to explore the effectiveness of the implemented improvements to the CAR process. Specifically, the asset discovery and vulnerability assessment aspects of the tool will be scrutinised. Additionally, this chapter will explore how the developed solution compares to current developed tooling in terms of features and capabilities. Finally, this chapter will discuss the need for CAR tooling and delve into avenues for future work, including suggestions for any possible improvements that can be made to further improve the effectiveness of CAR tooling in the future.

5.2 INTERNET-WIDE SCANNING PLATFORMS

By reviewing and critically analysing related literature, it was depicted that IWS platforms play a fundamental role within the cybersecurity ecosystem in terms of both offensive and defensive capabilities. The abundance of research surrounding IWS platforms highlighted several flaws and drawbacks associated with relying on such information, with the fluctuating timeliness of scanning frequency being a key detrimental shortcoming across all IWS platforms (Li *et al.,* 2021). Furthermore, scanning capabilities, reach and limitations were found to differ vastly between platforms, leading to the conclusion that several platforms should be utilised in unison and combined in order to obtain a dependable and comprehensive stream of data from the sources. Such an approach was speculated to be capable of mitigating IWS related issues identified in literature and was a unique aspect considered by this project.

Of the six IWS platforms mentioned within the literature review, only four were selected for integration into Informant to obtain a reliable and comprehensive stream of IWS data. The two discarded platforms were deemed inadequate for this project primarily due to their Chinese origin and related language barrier. Of the four IWS platforms, only the free and academic tiers of Shodan, Censys, and BinaryEdge could be leveraged to perform asset discovery via a CIDR search. The other selected platform, Onyphe, was the only platform found to require a paid monthly subscription to utilise the CIDR search feature, limiting the full potential of this platform. This feature limitation turned out to be a continuous nuisance, resulting in difficulties when comparing the detection rates of IWS platforms, as shown in Figure 9. For this comparison, Onyphe had to be recursively searched 256 times for every potential IPv4 address within the /24 block scanned, requiring the query load to be spread across two separate accounts not to exceed the monthly limit of 250 IP queries. This feature limitation ultimately resulted in the Onyphe platform only being utilised as an asset data enrichment source rather than a discovery source in the final Informant tool. Additionally, to not exceed the Censys free account credit limitation of 250 queries a month, only a

limited selection of data on each asset detected is considered by Informant. The full potential search capabilities of BinaryEdge and Shodan were able to be fully fulfilled by Informant due to the relatively non-restrictive academic Shodan account and free allocation of additional BinaryEdge credits. However, even though the full potential of Shodan and BinaryEdge could be leveraged, due to time constraints, only required attributes were processed by Informant.

As previously stated, the process of utilising and automatically consolidating multiple IWS platforms is a unique feature of the developed Informant tool, aimed at ensuring the most reliable and up to date picture of an exposed attack surface can be retrieved. Scout and ShoVAT, two comparable CAR tools that solely rely on one IWS platform as a source of information, are highly susceptible to the adverse effects of timely scanning frequencies, platform blacklists and credit limits (Shori, 2018; Wan *et al.,* 2020; Li *et al*., 2021). The specific issue of blocked scans is especially prevalent in the case of Scout due to its dependence on Censys, which is known to be blocked by three large hosts, accounting for 4% of all total hosts accountable within the IPv4 namespace (Wan *et al.,* 2020). Significantly impacting the capabilities of Scout. Informant mitigates all the issues stated above by using several sources, which also comes with other advantages, such as the ability to still use the tool in the event of critical issues with platforms such as the recent multiple-day outage of the Onyphe service (Onyphe, 2019). Overall, Informants ability to support multiple IWS platforms increases the tool's accuracy, versatility and reliability drastically. This enables the tool to build a more accurate and concise picture of an exposed attack surface than both Scout and ShoVAT (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019).

5.3 Capabilities of Developed Solution

During the example operation, a showcase of the capabilities Informant possesses were demonstrated on a subsection of the Abertay network range, in addition to a test run across four other anonymised UK universities. From this stage, interesting BinaryEdge, in all cases, detected the most assets out of the four IWS platforms, contrasting earlier results obtained within the initial IWS comparison section in the literature review. Whereby, Censys was identified in the isolated investigation to detect the most unique IP addresses. This larger test concluded that the BinaryEdge platform had the best scanning capabilities based on the five universities examined, with the presumed reasoning behind such thought to be down to several factors, including the range of scanned ports, IP blacklists and platform scanning frequencies. However, additional experimentation is required across a larger sample set spanning several industries to accurately determine which platform, on average, possesses the best scanning detection rate and why – such additional research is out of this project's scope. Nonetheless, from the research reviewed and results gathered, it is expected that not one IWS platform will dominate in all industry scenarios. Of the several features highlighted by the example operation, one of the most notable is an insight into the usefulness of the IWS platform merging process, enabling enrichment of individual IWS data sources. Data from all four IWS platforms can be seen to improve the effectiveness of the CAR process, vastly improving the accuracy and relevance of data held on assets, and comparable to active scanning techniques. Although the CAR process may never be able to compare in terms of data reliability in comparison to active scanning techniques, the merging process adopted is a considerable step in the right direction. Vastly improving the overall reliability of IWS data received. However, a potential drawback of the merging process is that several IWS platform accounts are required to be maintained, with various credit limits, potentially leading to extra financial costs if several paid accounts are enrolled.

In line with the research questions, several non-active additional reconnaissance sources were considered to explore how further non-active reconnaissance methods could assist with identifying threats during the literature review. Of the sources reviewed, PDNS was selected as the most appropriate solution. This resulted in six PDNS reconnaissance sources being supported by Informant which aided with creating a visually appealing interactive exposed attack surface network map. Such node-based graphs are great at making the sheer quantities of DNS and asset information quantifiable and are often seen as a stepping-stone in the journey from data to decision making (Singh, 2019). As identified in the literature review, further enriching this data allowed it to identify other potential threats known as dangling DNS records. Additionally, the obtained PDNS data allowed for related outside assets, such as assets hosted on cloud environments, to be identified and highlighted, furthering the asset discovery capabilities of Informant and improving the overall effectiveness of the tool.

Informants support for the merging of multiple IWS platforms and its stealth non-interactive nature makes it a great tool to deploy in scenarios where passive scanning is ineffective and intrusive active scanning techniques are deemed abrupt. From research examined, active scans have shown to crash fragile ICS devices and PLC services as well as hamper network throughput, all factors that have adverse effects on performance (Ferretti, Pogliani, and Zanero, 2019; Hashida, Kawamoto and Kato, 2020). Additionally, passive scans have the drawback of requiring the target to have active networking activity as well as the need for the strategic placement of monitors. These factors make tools adopting the CAR approach, such as Informant, a great alternative to these two traditional solutions.

The developed tool is well suited towards offensive scenarios where discretion is critical, such as in red teaming exercises or cyberwar games. The clean user interface and suite of automated graphics generated by the tool proved valuable in spotting potential security issues not picked up by integrated automated risk checks. For example, the generated graph in Figure 72 allowed for the detection of an exposed single factor Vivotek camera system located behind basic HTTP authentication to be revealed from the data. If required to be exposed to the internet, such an asset should be protected via more secure means. This result illustrates the type of security flaws that Informant and other CAR tools do not currently automatically consider, with such detection currently requiring active based scanning tools. The additional rDDoS, high-risk port list and PDNS enrichment features of Informant all proved to be excellent at providing extra insight into the security status of assets. Each proving to be sufficient at improving the overall effectiveness of the CAR process.

The testing across the four anonymised universities identified that 50% of all assets processed were vulnerable, with 5515 CVEs being identified across 953 services, of which 48% were categorised as vulnerable by default on their base configuration. Although not directly comparable, O'Hare, Macfarlane and Lo (2019) drew similar conclusions across a slightly larger but similar dataset, finding 12967 CVEs across seven UK-based academic institutions. Based on this data and the results gathered from Informant, it can be concluded that on average academic institutions within the UK are much less susceptible to known-vulnerabilities now than they were in 2018 – the year in which Scout performed its experiments. Additionally, a similar CVE CVSS spread of 5 across the educational institutions analysed was identified, which is in line with the conclusion drawn by Genge and Enăchescu (2016) and from further research conducted by O'Hare, Macfarlane and Lo (2019).

5.4 Internet-wide Scanning Data Processing

Experiment 1 set out to evaluate the effectiveness of merging information gathered from multiple IWS platforms using a series of five pre-configured servers. The results concluded that IWS data from multiple platforms were used to produce merged records across each of the five pre-configured servers, proving that utilising several sources of IWS data significantly helps with data reliability. A common issue faced by the reviewed CAR tooling relying on single IWS platforms. This improved reliability, in turn, increased the accuracy of service banner data used for known-vulnerability assignment, hence increasing the overall reliability and accuracy of the tool.

Although this experiment was a success, it was considerably resource intensive and time-consuming due to the need to deploy servers several weeks prior to experimentation, ensuring each IWS platform had adequate time to index the servers fully. This highlights a key problem, as even though Informant uses multiple IWS platforms, it does not entirely eradicate the negative effects of the timelessness of platform scanning frequencies. However, the issue is severely mitigated.

From the experiment conducted, BinaryEdge was found to be capable of gathering the most information on assets; however, the data was found not always to be the most recent. Analysing the results breakdown, each of the IWS platforms appear to be equally competitive regarding the frequency of scans, as illustrated by the even attribute selection spread in Table 15. This even attribute selection spread makes it incredibly difficult to pinpoint a sole platform as the most optimal due to significantly differing scanning frequencies.

5.5 Unique Identifier Construction

Experiment 2 evaluates the accuracy of the CPE construction feature implemented via manual and comparative assessment, with the related OpenVAS CPE construction feature being used for comparison. For this experiment, the well-considered methodology employed by O'Hare, Macfarlane and Lo (2019) to evaluate Scout was replicated. This process involved analysing and comparing 29 automatically constructed CPE banners against their respective true CPEs, as determined by the NVD. This experiment used a combination of non-obfuscated – banners which included version information – banner data from the example operation and previous experiment, ensuring a diverse non-bias range of banners were selected. The one false positive result identified during this experiment was found to be due to an unexpected design limitation within the implemented CPE construction algorithm. A failure to consider operating system components was found to be the direct cause of the false positive, as Informant was only designed to reproduce application based CPEs. This error was unexpected as it was wrongly assumed that service banners would only be a type of application during the design phase, not a type of operating system or hardware. These alterative part types are denoted by a CPE part value of 'o' and 'h', respectively. Although, from the data, it is evident that only in rare instances do IWS platforms return banner information classified as anything other than an application. Hence, this issue has minimal effect on the overall performance of the CPE construction algorithm but is something to consider in future work.

Two false negative errors during this experiment were discovered to be a result of significantly differing CPE values in comparison to the banner information obtained from the IWS platforms. Such a discovery identifies an issue with Informants reliance on the Levenshtein algorithm for CPE validation, as if garbage

complex data is inputted, garbage will be returned. Although the set ratio threshold and exception list were developed to help control and mitigate this problem, in these two cases the mitigations failed. Investigating the false negatives, it appeared that they were caused due to unconventional CPE naming conventions, which can be attributed to a lack of service banner regulation for software developers. Ultimately, this investigation identified these issues to be associated with the shortcoming of the NVD, supporting the conclusion of work by O'Hare, Macfarlane and Lo (2019).

Comparing and evaluating Informants CPE construction accuracy against Scout showed that the adopted CPE construction process helped Informant outperform Scout across all comparable evaluation metrics. The extra available information-rich merged data was also a contributing factor to why Informant outperformed the competition. For the comparative assessment against OpenVAS, the preconfigured servers used in experiment 1 were utilised for testing as the timeline of this project did not allow for additional server deployment; largely in part due to the time required for IWS platforms to index all services. The obstacle of obtaining permission for using these tools was averted by utilising a personal dedicated server, allocating additional IPv4 addresses and slicing it up using virtualisation software in a bid to ensure no firewalls or other cloud-related security infrastructure obscured the results.

The results from the comparative aspect of this experiment surprisingly found that Informant was more capable of accurately generating CPEs than OpenVAS. This result was unexpected as more detailed service information is available to OpenVAS due to its active scanning approach, theoretically making the construction of CPEs easier. However, additional OS meta-data within service banners appear to hamper the ability of OpenVAS to return a completely accurate CPE, whilst Informant managed to handle these scenarios correctly due to the inclusion of a service exceptions list and the use of the Levenshtein algorithm. The results from this experiment are corroborated by O'Hare, Macfarlane and Lo (2019), who found Scout, which utilises a similar CPE construction approach, to also be likely to be more accurate than OpenVAS. Additionally, the authors Genge and Enăchescu (2016) also support this conclusion, finding a competing enterprise product, Nessus, unable to produce CPEs from complex banners correctly. Although the Scout tool was not directly comparable due to its restrictive protocol support, from the scientifically quantified results, Informant can be seen to have a greater precision than Scout by a factor of 0.01. This can be argued to be down to the more refined CPE construction approach adopted by Informant. Lastly, the ShoVAT comparison found Informant to have a better precision and a lesser chance than ShoVAT of producing a false positive result. This point is often an impactful decisive factor when selecting a security tool.

5.6 KNOWN-VULNERABILITY ASSIGNMENT

The third experiment consisted of two sections, the first aimed at evaluating the CVE assignment process against OpanVAS. The second aimed at determining the usefulness and accuracy of the vulnerable by default base configuration identification process of Informant. The initial experiment required the reconfiguration of two of the previously configured servers used in past experiments, in order to scan them with OpenVAS. Due to project time constraints and the scanning lag associated with IWS platforms (Li *et al.,* 2021), for this experiment, Informant service banner data was manually manipulated to represent the configuration of the servers scanned by OpenVAS. Such manipulating is possible in Informant without hindering the integrity or accuracy of the results. Additionally, Scout was also considered for the comparative process. However, as it employs an identical approach to CVE assignment

as Informant and has minimal support for protocols, it was excluded. Nessus was also considered but excluded due to CVE assignment issues.

From the analysis of the first section of experiment results, OpenVAS was observed to avert false positives better than Informant due to additional validation checks made possible as a result of active scanning techniques. This result highlights a key downside to the CAR process that Informant employs. The results also unexpectedly indicate that Informant in three of the four test cases could not assign the same amount of vulnerabilities as was contained within the NVD. An unusual revolution considering that Informant uses CVE data from the NVD. Hence, it was expected that the results from both should match. After further investigation, this somewhat contradictory result was found to be related to the local copy of the NVD obtained via the cve-search utility having inconsistencies within the CPE to CVE data correlation. A possible explanation for these issues is likely related to inconsistencies and errors found throughout the NVD dataset (Anwar *et al.,* 2020). Similar issues plagued the Scout tool with O'Hare, Macfarlane and Lo (2019) resulting in a similar conclusion.

The unique vulnerable by default configuration flagging process attempted to use data enrichment to highlight CVEs that could be exploited on the related services default configuration. The evaluation of this feature found Informants performance to be subpar against the tested Nginx services and it was only found to have an accuracy of 67% against OpenSSH services. However, due to the inconsistent nature of the NVD and the small test sample considered, these results should be interpreted with caution. Although the results are not very encouraging, a key takeaway from this experiment is that the considered CVSS vector components do not hold enough weight or contain enough data to classify CVEs as vulnerable by default on their own accurately. It is speculated that by implementing CVE summary text-mining in conjunction with the adopted vector approach, the accuracy could be significantly improved, lowering both false positive and false negative rates drastically. Nonetheless, the implemented solution helps point out high-risk known-vulnerabilities amongst the crowd of assigned CVEs, whilst outperforming the CVE assignment process of Scout (O'Hare, Macfarlane and Lo, 2019).

5.7 TOOL COMPARISON

The fourth and final experiment comparatively assessed the functionality of Informant against similar tooling. The results found that Informant was vastly more feature rich than competing CAR tools in existence, with it being the only known CAR tool found in literature to be user-friendly and feature a web interface. In comparisons to Scout and ShoVAT - the latter of which is closed source - the developed tool was found to outperform both when it came to accuracy, useability, and features. Further comparing the tool against industry-standard tooling it was observed that Informant could compete well against the features and capabilities provided by traditional active and passive scanning-based tools. Although, it is important to bear in mind that the previously drawn upon limitations of the CAR process apply to Informant. Notably, the inability to gather near real-time data on devices, which is something active scanning tools can achieve with ease, all be it at the sacrifice of stealth.

The developed tools ability to compete with industry-standard tooling without directly communicating with target devices is by far the best outcome for this project. Informant was found to produce comparable results to professional, often expensive tooling solutions whilst achieving an unmatched level of anonymity. The tools open-source nature and user-friendly interface also makes the tool more

appealing than the competition, designed from the ground up to be usable with no prior knowledge or experience. Although, not a scientifically evaluated metric within this study, the overall performance of Informant in comparison to traditional network assessment tools could be seen to be drastically better. This result reflects findings by Genge and Enăchescu (2016), who found their CAR tool, ShoVAT, to have a significantly faster execution time compared to Nessus, a comparable traditional network scanning tool to OpenVAS. The observed increase in performance in both cases can be related to there not being a need to perform direct time-consuming active scanning. Highlighting an advantage of the CAR process, as the IWS platforms handle all the time-consuming scanning processes.

5.8 Limitations and Future Work

Due to this project's highly restrictive time constraint and ambitious objectives, certain aspects of the project were narrowed as time progressed. Thus, allowing for multiple areas of future work to explore. Additionally, distinct limitations can be found within the developed Informant tool, primarily in part to the academic origin of the tool. These limitations can affect the tool in scenarios out with the conditions in this paper. However, the developed Informant tool will be made open-source and available for public contributions in due course.

Additional IWS, PDNS and other OSINT data sources could be integrated into the tool, adding to the already available four IWS and six PDNS sources. Unfortunately, it was not plausible to achieve this in the time available to undertake this project. However, such extra sources would almost certainly increase the quantity and integrity of asset information, hence improving the CPE construction algorithm's accuracy, leading to more precise known-vulnerability assessments. Additionally, limitations of the already integrated IWS and PDNS platforms could be addressed to improve the tool's efficiency by taking full advantage of each data source. Furthermore, Informant currently does not consider all data ingested by the integrated data sources, primarily due to project time constraints. Leaving much room for future expansion to the current data sources integrated. The efficiency of the current integration of the IWS platforms could also be further optimised for paid accounts, as this project was based on free and academic accounts; no such consideration for paid IWS platform API features has been made. Specifically, the current Censys implementation significantly limits the amount of obtainable information as it is designed and optimised to consume the least amount of credits possible, allowing for the project experiments to be conducted.

Future advances in NVD research could improve the CPE construction algorithm's overall efficiency due to a more profound understanding of the NVD's shortcomings. Taking inspiration from work by Wåreus and Hell (2020), in future investigations, the CPE construction methodology could be replaced with a more innovative NLP approach, which may provide better, more accurate CPE predictions. Many of the issues and inconsistencies found within this project can be related to the poor state of the NVD. Therefore, future work could explore alternatives to the NVD, or even investigate the possibility of merging several vulnerability databases into one consolidated, reliable dataset. Additional non-traditional means of vulnerability data sources could also be considered in future work, similar to how CSTOOL (2021), a non-academic software as a service solution correlates news articles to obtain CVE data.

The idea of analysing CVEs to look for known-vulnerabilities that are vulnerable by default configurations was found to be a more complex problem to solve than first anticipated. As a result, a further study

focusing on this area, investigating how text-based summary mining in conjunction with CVSS vector analysis could improve this process is suggested. In hindsight, considering the lessons learned from this study, more focus would have been directed towards the proposed text-based summary mining default CVE identification approach suggested, than the implemented CVSS vector approach. Nonetheless, the implemented and proposed approach is a good starting point for future studies in this area, which is based on work by the authors Guo, Xing and Li (2020) and work by Wåreus and Hell (2020).

To further evaluate the functionality and usefulness of this tool in the real world, a more comprehensive study of the tool across sectors spanning beyond educational institutions could be conducted. A longitudinal study could also be performed to allow for an insight into how intuitive the tool is for end users over long durations. However, this would require multiple users to adopt and maintain a local copy of the tool, which was out of this project's scope. Additionally, further experiments could have been conducted investigating the effectiveness of specific aspects of the tool in more detail, such as the developed fraudulent geolocation verification, rDDoS prediction and automated interactive network mapping features of the tool. Although these features were implemented, they were additions to the project as it progressed and were not initially planned for; hence, they were not factored into the original time frame allocated for experimentation.

Moreover, a comparative assessment comparing how broad of an attack surface Informant can enumerate on a given network range against industry-standard tools such as OpenVAS and Nessus would be of interest. Unfortunately, ethical constraints surrounding obtaining permission to actively scan an entire network range prevented such an experiment in this study, leaving room for further studies evaluating the tool.

During the development of this project, attack surface monitoring and attack surface management platforms began to gain traction in the broader security community. Censys, Shodan and BinaryEdge are all currently pushing their own enterprise solutions. Additionally, two emerging open-source and commercialised solutions, Spiderfoot (2021) and Intrigue (2021), also became well known during this study, with Intrigue successfully secured a two-million-dollar investment in April of 2021 (Intrigue, 2021a). These recent emerging platforms, along with Informant, clearly highlight a gap in the market and room for future innovation within the IWS and CAR sector. In future, given more time, a more novel and refined version on Informant could be implemented.

5.9 CONCLUSION

Overall, the experiment results provided a great insight into how well the developed solution performed compared to other tooling found within literature and industry. Continuous improvements implemented to the tool throughout the design and implementation phases of the project were brought about by issues encountered, which ultimately helped improve the overall efficiency of the tool. Many of the procedures and scientific evaluation methods used in the experimentation phase were based on the evaluation of Scout (O'Hare, Macfarlane and Lo, 2019), which expanded upon the methods used to evaluate ShotVAT (Genge and Enăchescu, 2016). The virtualisation of a sole private dedicated server was a great solution for setting up test servers for use within the experiments, avoiding the extra expense of acquiring additional cloud resources. However, obtaining additional IPv4 addresses required manual justification

and approval by the datacentre, a tedious and time-consuming process. A factor which cloud solutions could have averted, but at a more significant financial expense.

Informants' broad feature set and non-interactive nature set it apart from the related tooling compared within this study, with the tool being primarily targeted toward applications in defensive and offensive cybersecurity. The tools free open-source and user-friendly approach helps ensure it is adoptable by a broad audience, regardless of technical skill, an advantage over other command line-based CAR tools found in literature. The unique integration of the predicted total rDDoS BAF and CVEs vulnerable by default configuration further improves the practicality of the tool. Furthermore, the user project-based design and historical time-series approach to data storage, allow for the management and coexistence of multiple projects, enabling swift asset discovery and security assessment across projects. Bug bounty hunters, red teams and blue teams can best utilise the tool to help identify low hanging fruit because of Informants ability to identify a comprehensive view of a given Internet-facing attack surface.

CHAPTER 6

6 CONCLUSIONS

This project aimed to produce and evaluate a non-intrusive asset discovery and vulnerability assessment tool capable of identifying known-vulnerabilities in exposed Internet-connected devices. The development of a novel tool, Informant, was produced to achieve this aim. Informant accomplishes the project objectives by retrieving freely available data from four IWS platforms and six PDNS sources, effectively correlating and processing the data to construct a picture and risk assessment of a given network's Internet-facing attack surface. The tool employs and improves upon a well proven approach of performing known-vulnerability assessment by utilising data from the NVD (Genge and Enăchescu, 2016; Sanguino and Uetz, 2017; O'Hare, Macfarlane and Lo, 2019). In addition to employing further techniques and sources from literature to enhance the tool's overall discovery and risk assessment effectiveness. The following chapter outlines the conclusions which can be drawn from this study.

A comprehensive review of the literature identified several key findings. By examining and comparing literature surrounding IWS platforms, several well-established and emerging use cases of IWS data in the cybersecurity realm were identified (Durumeric, Wustrow, and Halderman, 2013; Durumeric *et al.,* 2015; Leverett and Kaplan, 2017). During the literature analysis, four appropriate IWS platforms were identified as suitable for data ingest into Informant, all of which were found to identify and enumerate service banner information efficiently (Bodenheim *et al*., 2014; Durumeric *et al.,* 2015). Two alternative Chinese-based platforms analysed during this stage were discounted due to a lack of research and evidence of businesses singling out China, blocking large associated network ranges. Such blocking adversely affects overall scanning effectiveness of platforms (Matherly, 2014). In addition, ongoing political turmoil between China and western democracy has the potential to further negatively influence the performance of these platforms. Through a further in-depth review and investigation of the four selected platforms, Shodan, Censys, BinaryEdge and Onyphe, the shortcomings of solely relying on one IWS platform for accurate information was apparent. It was determined that results from IWS platforms vary for numerous reasons, such as IP blacklisting, scanning frequencies and network-based security systems (Shori, 2018; Li *et al.,* 2021). This research analysis led to the consensus that merging multiple sources would allow for a more reliable picture of assets to be obtained, by mitigating the impact of the previously mentioned issues surrounding IWS platforms.

Within the literature review, current vulnerability assessment tooling utilising active, passive and CAR approaches were explored. Existing passive vulnerability assessment tooling, although non-intrusive, were found to significantly lack capability, functionality, and usability compared to their active scanning counterparts, justifying the aim of this project. However, although more comprehensive and effective at detecting known vulnerabilities, active scanning vulnerability assessment tooling is known to be intrusive and disruptive to target networks and devices (Hashida, Kawamoto and Kato, 2020). Thus, providing further justification for the project, as the CAR process to be adopted does not directly interface with targets, a drawback of active scanning. Furthermore, three tools within literature were found to adopt the CAR approach, of which none were found to be user-friendly, and all were found to only consider data

from one IWS platform (Genge and Enăchescu, 2016; Rodriguez *et al.,* 2017; O'Hare, Macfarlane and Lo, 2019). Which, as previously discovered from literature, is likely to allow numerous assets to go undetected. Informant set out to mitigate all the pitfalls of current CAR tooling mentioned above.

Analysing literature surrounding two current CAR known-vulnerability assessment tools, Scout and ShoVAT, helped influence critical aspects of Informants design within the methodology section of this project. During the literature review, several papers indicated issues with inconsistencies within the NVD dataset, with the most troublesome concern being the presence of a lengthy lag in the reporting of vulnerabilities (Rodriguez *et al.,* 2017), diminishing the accuracy of VMS and related tooling relying on this data. Scouts CPE construction approach, formed using a blend of prior techniques, primarily based on work by Sanguino and Uetz (2017), was identified as being able to mitigate most of the identified inconsistency issues within the NVD. As a direct result of this, it was selected as the base design for the Informant's CPE construction algorithm implementation. In addition, this project further refined the CPE construction approach, factoring in an additional service exceptions list and weighted ratio for the Levenshtein distance algorithm compared to the traditional basic edit distance approach used by Scout.

Interpreting further literature looking at additional avenues that OSINT data could contribute to improving the effectiveness of the CAR process found PDNS to be an overlooked outlier out of the sources reviewed. From literature, the use case of PDNS was predominantly found to be within the realm of malicious domain name analysis (Xuanzhen, Zulie and Yuanchao, 2020). However, a lack of research surrounding the use case of PDNS in identifying dangling DNS records was discovered, identifying a gap in current research, which was selected to be explored via implementing this data source into Informant. This feature further expanded the asset discovery and risk assessment functionality of Informant beyond that of the comparable Scout and ShoVAT tools found in current literature (Genge and Enăchescu, 2016; Sanguino and Uetz, 2017; O'Hare, Macfarlane and Lo, 2019). Enabling Informant to identify associated organisation assets out with the initially provided network range, such as devices located on the cloud.

The implemented tool was tested and evaluated by means of experimentation, in line with the produced methodology. Most of the experiments within this paper followed well-established evaluation techniques, based on previous work by O'Hare, Macfarlane and Lo (2019). An example operation in addition to six distinct experiments was performed to achieve this. One focusing on the effectiveness of the IWS merging process, two on the CPE construction process, one investigating the assignment of CVEs, another exploring CVEs vulnerable by default and another comparing the developed solution with related tooling. The scientific measurements used to quantify results within the both CPE and CVE assignment experiment increased the reproducibility of results. The metrics also allowed for direct comparisons to be made with similar tooling, notably OpenVAS and Scout.

The experiments conducted revealed several key findings, with Informants several IWS data sources underpinning the project's core foundation. The first experiment found the merging of IWS data to help drastically increase the reliability and integrity of data, resulting in the tools improved ability to derive a more conclusive and accurate exposed attack surface compared to competing CAR tooling, ShoVAT and Scout (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019). Across all assets merged during experimentation, multiple attributes from several IWS platforms were used to create every merged record, proving the effectiveness and usefulness of the feature. The advantages of this feature were found to go beyond allowing for better asset and attribute detection rates, with the feature also found to improve upon the versatility and redundancy of the tool vastly. These are traits that similar tools lack to

fulfil in any manner. Using the findings from the experiments and observing the differences in data obtained from each platform, the data can be seen to back up existing knowledge surrounding IWS platform probes being intentionally obstructed by devices.

Experiments evaluating the known-vulnerability capabilities of Informant found the tool to surpass the accuracy of ShoVAT and Scout when it came to the construction of CPE banners. This high level of precision can be associated with the unique adaptions made to the adopted approach to CPE construction, which is based on previous research, taking into considerations inconsistencies within the NVD (Sanguino and Uetz, 2017; O'Hare, Macfarlane and Lo, 2019). This result led to the consensus that Informant has a better overall known-vulnerability assignment rate than Scout and ShoVAT due to the production of more accurate CPEs. Additionally, Informants processing capabilities far exceeded that of Scout, with no limitation on ports or services. Further comparisons with industry tooling observed that OpenVAS could avert false positives better than Informant due to additional validation checks made possible due to active scanning techniques. Highlighting a key limitation of the CAR process. To further improve upon the CPE construction process in future work, it is suggested that an NLP machine learning approach is adopted to perform CPE construction, based on research by Wåreus and Hell (2020). Adapting such an approach to work on service banner data is expected to yield interesting results.

The unique feature designed to flag CVEs vulnerable by default configuration was found to have an accuracy of 67% across two specific OpenSSH services, by just using the implemented CVSS vector approach methodology. Ultimately this implemented solution helps successfully showcase high-risk known-vulnerabilities amongst a crowd of assigned CVEs, whilst outperforming the CVE assignment process of Scout. Whilst this feature of Informant gave an insight into the area of default CVE configuration identification, it leaves many avenues for further improvement. Notably, due to time constraints the proposed text-mining approach for identifying CVEs vulnerable by default could not be fully implemented or researched further. Integration of such an improvement is believed to be capable of drastically improving the accuracy of the vulnerable by default detection feature and is a great avenue for future work to explore.

The final experiment comparatively assessed the functionality of Informant against related developed tooling, finding Informant to outperform both Scout and ShoVAT when it came to accuracy, useability, and feature sets (Genge and Enăchescu, 2016; O'Hare, Macfarlane and Lo, 2019). The developed tool was also found to compete well against traditional industry-standard vulnerability assessment tooling, surpassing the risk identification capabilities of all current free non-active features of the reviewed tools in this paper, including OpenVAS and Nessus.

Overall, the developed tool helped effectively improve various aspects of the CAR process and build upon the surrounding capabilities of associated tooling. Exploring and collating numerous forms of data indirectly to formulate a tool with a high level of effectiveness. The final solutions non-active nature allows it to perform comprehensive asset discovery and risk assessments on networks without leaving a trace, making the tool an excellent solution for offensive security teams and bug bounty hunters performing stealth reconnaissance tasks. The example operation gave an insight into the potential of the tool by identifying 5,515 known-vulnerabilities across 4 UK-based universities, of which 48% were found to be vulnerable by default. This study has set a standard for exploring various ways the CAR process can be expanded upon to produce a non-evasive tool capable of competing with enterprise-grade solutions, leaving and introducing several future avenues to build upon and explore.

REFERENCES

Anonymous. (2012) *Internet Census 2012*. Available at: http://census2012.sourceforge.net/paper.html (Accessed: 27th January 2021).

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K. and Zhou, Y. (2017) Understanding the Mirai Botnet. 'Understanding the Mirai Botnet', *SEC'17: Proceedings of the 26th USENIX Conference on Security Symposium*, Vancouver, Canada, pp. 1093 – 1110.

Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou, N. and Dragon, D. (2011) Understanding the Mirai Botnet. 'Detecting malware domains at the upper DNS hierarchy' *SEC'11: Proceedings of the 20th USENIX Conference on Security*, San Francisco, USA.

Anwar, A., Abusnaina, A., Chen, S., Li, F. and Mohaisen, D. (2020) 'Cleaning the NVD: Comprehensive Quality Assessment, Improvements, and Analyses', *arXiv.*

Bachmann, M. (2021) *RapidFuzz 1.0.0 documentation.* Available at: https://maxbachmann.github.io/RapidFuzz/fuzz.html (Accessed: 27th March 2021).

Bano, S., Richter, P., Javed, M., Sundaresan, S., Durumeric, Z., Murdoch, S., Mortier, R. and Paxson, V. (2018) 'Scanning the Internet for Liveness', *SIGCOMM Comput. Commun. Rev*, 48(2), pp. 2-9. doi: 10.1145/3213232.3213234.

Bartlett, G., Heidemann, J. and Papadopoulos, C. (2007) 'Understanding Passive and Active Service Discovery', *7th ACM SIGCOMM conference on Internet measurement*, San Diego, California, pp. 57-70. doi: 10.1145/1298306.1298314.

Bilge, L., Sen, S., Balzarotti, D., Kirda, E. and Kruegel, C. (2014) 'Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains', A*ssociation for Computing Machinery,* 16(4). doi: 10.1145/2584679.

BinaryEdge. (2021) *BinaryEdge.* Available at: https://www.binaryedge.io/ (Accessed: 28th January 2021).

BinaryEdge. (2021a) *BinaryEdge API Documentation.* Available at: https://docs.binaryedge.io/ (Accessed: 28th January 2021).

BinaryEdge. (2021b) *Product - BinaryEdge.* Available at: https://www.binaryedge.io/data.html (Accessed: 28th January 2021).

BinaryEdge. (2021c) *Pricing - BinaryEdge.* Available at: https://www.binaryedge.io/pricing.html (Accessed: 28th January 2021).

Bodenheim, R. C., Butts, J., Dunlap, S., & Mullins, B. (2014). Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices. *International Journal of Critical Infrastructure Protection*, 7(2), pp. 114-123. doi: 10.1016/j.ijcip.2014.03.001.

Borbolla, R. (2019) *Tools for OSINT*. Available at: https://renatoborbolla.medium.com/tools-for-osint-cf5a86cede67 (Accessed: 12th February 2021).

Borges, E. (2019) *Top 20 and 200 most scanned ports in the cybersecurity industry.* Available at: https://securitytrails.com/blog/top-scanned-ports (Accessed: 18th February 2021).

Bou-Harb, E., Debbabi, M. and Assi, C. (2014) 'Cyber Scanning: A Comprehensive Survey', *IEEE COMMUNICATIONS SURVEYS & TUTORIALS*, 16(3), pp. 1496-1519. doi: 10.1109/SURV.2013.102913.00020.

Brewster, T. (2018) *A Hacker Forced 50,000 Printers To Spread PewDiePie Propaganda -- And The Problem Is Much Bigger Than You Know* . Available at: https://www.forbes.com/sites/thomasbrewster/2018/12/03/a-hacker-forced-50000-printers-to-spread-pewdiepie-propagandaand-the-problem-is-much-bigger-than-you-know/?sh=4aee28f13819 (Accessed: 21th January 2021).

Censys. (2020) *Expanded List of New Lightweight IPv4 Port Scans for Enterprise Customers*. Available at: https://support.censys.io/hc/en-us/articles/360038755571 (Accessed: 28th January 2021).

Censys. (2021) *Censys*. Available at: https://censys.io/ (Accessed: 1st December 2020).

Censys. (2021a) *Censys Product Introduction*. Available at: https://support.censys.io/hc/en-us/articles/360038757271-Censys-Product-Introduction (Accessed: 12th February 2021).

Censys. (2021b) *Research Access to Censys Data*. Available at: https://support.censys.io/hc/en-us/articles/360038761891-Research-Access-to-Censys-Data (Accessed: 16th February 2021).

Censys. (2021c) *Expanded List of New Lightweight IPv4 Port Scans for Enterprise Customers*. Available at: https://support.censys.io/hc/en-us/articles/360038755571 (Accessed: 16th February 2021).

Censys. (2021d) *Censys REST API*. Available at: https://censys.io/api (Accessed: 16th February 2021).

Censys. (2021e) *Search API*. Available at: https://censys.io/api/v1/docs/search (Accessed: 26th March 2021).

Cheikes, B., Waltermire, D and Scarfone, K. (2011) *Common Platform Enumeration: Naming Specification Version 2.3*. Available at: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7695.pdf (Accessed: 22th February 2021).

Cimpanu, C. (2020) *University of Utah pays $457,000 to ransomware gang*. Available at: https://www.zdnet.com/article/university-of-utah-pays-457000-to-ransomware-gang/ (Accessed: 28th March 2021).

CISA. (2019) *UDP-Based Amplification Attacks.* Available at: https://us-cert.cisa.gov/ncas/alerts/TA14-017A (Accessed: 10th March 2021).

Croak, M. (2019) *Google Interview: 25 Horses*. Available at: https://medium.com/@mattcroak718/google-interview-25-horses-c982d0a9b3af (Accessed: 10th March 2021).

CSTOOL. (2021) *CSTOOL.io – The cybersecurity toolbox.* Available at: https://attacksrfc.cstool.io/homepage.html (Accessed: 18th April 2021).

DataTables. (2021) *DataTables | Table plug-in for jQuery.* Available at: https://datatables.net/ (Accessed: 22nd February 2021).

Deraison, R. and Gula, R. (2009) *Blended Security Assessments: Combining Active, Passive and Host Assessment Techniques*. Available at: https://silo.tips/download/blended-security-assessments (Accessed: 21th January 2021).

Dirolf, M. and Bernie, H. (2021) *PyMongo 3.11.3 Documentation.* Available at: https://pymongo.readthedocs.io/en/stable/#overview (Accessed: 22nd March 2021).

Distance24. (2021) *Distance API.* Available at: https://www.distance24.org/api.xhtml (Accessed: 25th March 2021).

Django. (2021) *Web framework for perfectionists with deadlines*. Available at: https://www.djangoproject.com/ (Accessed: 10th March 2021).

Durumeric, Z., Adrian, D., Mirian, A., Bailey, M. and Halderman, J. (2015) 'A Search Engine Backed by Internet-Wide Scanning', *CCS '15: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 542-553. doi: 10.1145/2810103.2813703.

Durumeric, Z., Wustrow, E. and Halderman, J. (2013) 'ZMap: Fast Internet-Wide Scanning and its Security Applications', *USENIX Association,* pp. 605-620. doi: 10.5555/2534766.2534818.

Erb, S. (2019) *DNSGrep — Quickly Searching Large DNS Datasets.* Available at: https://blog.erbbysam.com/index.php/2019/02/09/dnsgrep/ (Accessed: 03rd March 2021).

Feldmann, A., Gasser, O., Lichtblau, F., Pujol, E., Poese, I., Dietzel, C., Wagner, D., Wichtlhuber, M., Tapiador, J., Vallina-Rodriguez, N., Hohlfeld, O. and Smaragdakis. (2020) 'The Lockdown Effect: Implications of the COVID-19 Pandemic on Internet Traffic', *Proceedings of the ACM Internet Measurement Conference*, New York, USA, pp. 1-18. doi: 10.1145/3419394.3423658.

Ferretti, P., Pogliani, M. and Zanero, S. (2019) 'Characterizing Background Noise in ICS Traffic Through a Set of Low Interaction Honeypots', *Proceedings of the ACM Workshop on Cyber-Physical Systems Security & Privacy*, pp. 51-61. doi: 10.1145/3338499.3357361.

FIRST. (2019) *Common Vulnerability Scoring System version 3.1: Specification Document*. Available at: https://www.first.org/cvss/specification-document (Accessed 27th March 2021).

Fitzgerald, W. and Foley, S. (2013) 'Avoiding Inconsistencies in the Security Content Automation Protocol', *2013 IEEE Conference on Communications and Network Security (CNS),* pp. 454-461. doi: 10.1109/CNS.2013.6682760.

Fjellskål, E. (2020) *Passive Real-time Asset Detection System.* Available at: https://github.com/gamelinux/prads (Accessed: 21st February 2021).

Flask. (2021) *Flask | The Pallets Projects.* Available at: https://palletsprojects.com/p/flask/ (Accessed: 21st February 2021).

Fleisher, C. (2008) 'Using open source data in developing competitive and marketing intelligence', *European Journal of Marketing*, 42(7), pp. 852-866. doi: 10.1108/03090560810877196.

Fofa (2021) *Fofa.* Available at: https://fofa.so/ (Accessed: 08th February 2021).

Frei, S., May, M., Fiedler, U., and Plattner, B. (2006) 'Large-scale vulnerability analysis', *LSAD '06: Proceedings of the 2006 SIGCOMM,* pp. 131-138. doi: 10.1145/1162666.1162671.

Genege, B., Haller, P. and Enăchescu, C. (2016) 'Beyond Internet Scanning: Banner Processing for Passive Software Vulnerability Assessment', *INTERNATIONAL JOURNAL OF INFORMATION SECURITY SCIENCE*, 4(3), pp. 81-91.

Genge, B. and Enăchescu, C. (2016) 'ShoVAT: Shodan-based vulnerability assessment tool for Internet-facing services', *Security and Communication Networks*, 9(15), pp. 2696–2714. doi: 10.1002/sec.1262.

GitHub (2021) *GitHub*. Available at: https://github.com/ (Accessed: 08th February 2021).

Goldman, D. (2013) *Shodan: The scariest search engine on the Internet*. Available at: https://money.cnn.com/2013/04/08/technology/security/shodan (Accessed: 08th January 2021).

Grangeia, L. (2019) *Data Insights on the BlueKeep Vulnerability*. Available at: https://www.bitsight.com/blog/data-insights-on-bluekeep-vulnerability/ (Accessed: 27th December 2020).

GreyNoise. (2021) *GreyNoise Visualizer.* Available at: https://greynoise.io/ (Accessed: 27th February 2021).

Gueye, B., Ziviani, A., Crovella, M. and Fdida, S. (2006) 'Constraint-based geolocation of internet hosts', *IEEE/ACM Trans*. Netw., 16, pp. 1219–1232. doi:10.1145/1028788.1028828.

Guo, H., Xing, Z. and Li, X. (2020) 'Predicting Missing Information of Vulnerability Reports', *WWW '20: Companion Proceedings of the Web Conference 2020,* pp. 81-22. doi: 10.1145/3366424.3382707.

Guy. (2020*) An Alternative to Shodan, Censys with User-Agent CensysInspect/1.1.* Available at: https://isc.sans.edu/forums/diary/An+Alternative+to+Shodan+Censys+with+UserAgent+CensysInspect11/26718/ (Accessed: 28th January 2021).

HackerTarget. (2019) *Brief History of Internet Wide Scanning.* Available at: https://hackertarget.com/remote-access-granted/ (Accessed: 28 December 2020).

Hanka, T., Niedermaier, M., Fischer, F., Kießling, S., Knauer, P. and Merli, D. (2021) 'Impact of Active Scanning Tools for Device Discovery in Industrial Networks', *Security, Privacy, and Anonymity in*

Computation, Communication, and Storage, Nanjing, China, pp. 557-572. doi: 10.1007/978-3-030-68884-4_46.

Hashida, H., Kawamoto, Y and Kato, N. (2020) 'Impact of Internet-Wide Scanning on IoT Data Communication in Wireless LANs'*, 2020 IEEE International Conference on Communications Workshops*, pp. 1-6. doi: 10.1109/ICCWorkshops49005.2020.9145257.

Hellard, B. (2020) *Hackers hold Newcastle Uni student data to ransom*. Available at: https://www.itpro.co.uk/security/ransomware/357022/hackers-hold-newcastle-uni-student-data-to-ransom (Accessed: 28th March 2021).

Henriques, T. (2015) 'BinaryEdge - Security Data Metrics and Measurements at Scale', BSides Lisbon.

Intrigue. (2021) *Intrigue - Enterprise Attack Surface Management*. Available at: https://intrigue.io/ (Accessed: 19th April 2021).

Intrigue. (2021a) *Intrigue Announces $2M Seed Round Funding*. Available at: https://intrigue.io/intrigue-announces-2m-seed-round-funding/ (Accessed: 20th April 2021).

Kandias, M., Mitrou, L., Stavrou, V. and Gritzalis, D. (2013) 'Which side are you on? A new Panopticon vs. privacy', *International Conference on Security and Cryptography (SECRYPT),* Reykjavik, Iceland, 2013, pp. 1-13.

Khadilkar, V., Rachapalli, J. and Thuraisingham, B. (2010) 'Semantic Web Implementation Scheme for National Vulnerability Database', Dallas: University of Texas.

LA-Shill. (2020) *Informant*. Available at: https://github.com/LA-Shill/Informant (Accessed: 17st April 2021).

Leverett, E. and Kaplan, A. (2017) "Towards estimating the untapped potential: a global malicious ddos mean capacity estimate," *Journal of Cyber Policy,* 2(4), pp. 195-208. doi: 10.1080/23738871.2017.1362020.

Leverett, E. and Rhode, M and Wedgbury, A (2020) 'Vulnerability Forecasting: In theory and practice', *arXiv.*

Li, R., Shen, M., Yu, H., Li, C., Duan, P. and Zhu, L. (2021) 'A Survey on Cyberspace Search Engines', *Communications in Computer and Information Science*, Beijing, China, pp. 206-214. doi: 10.1007/978-981-33-4922-3_15.

Lindome, P. (2020) *The SANS Guide to Evaluating Attack Surface Management*. Available at: https://www.sans.org/reading-room/whitepapers/analyst/guide-evaluating-attack-surface-management-39905 (Accessed: 13th March 2021).

Liu, D., Hao, S. and Wang, H. (2016) 'All Your DNS Records Point to Us: Understanding the Security Threats of Dangling DNS Records', *Association for Computing Machinery, pp. 1414–1425.* doi: 10.1145/2976749.2978387.

Lueth, K. (2020) *State of the IoT 2020: 12 billion IoT connections, surpassing non-IoT for the first time.* Available at: https://iot-analytics.com/state-of-the-iot-2020-12-billion-iot-connections-surpassing-non-iot-for-the-first-time/ (Accessed: 12th January 2021).

Manes, G., Schulte, D., Guenther, S. and Shenoi, S. (2005) 'NetGlean: A Methodology for Distributed Network Security Scanning', *Journal of Network and Systems Management*, 13, pp. 329-344. doi: 10.1007/s10922-005-6263-2.

Marchal, S., François, J., Wagner, C., State, R., Dulaunoy, A., Engel, T. and Festor, O. (2012) 'DNSSM: A large scale passive DNS security monitoring framework', *2012 IEEE Network Operations and Management Symposium*, pp. 988-993. doi: 10.1109/NOMS.2012.6212019.

Marzouk, Z. (2021) *University of Northampton hit by cyber attack*. Available at: https://www.itpro.co.uk/security/cyber-attacks/359003/university-of-northampton-hit-by-cyber-attack (Accessed: 28th March 2021).

Masscan (2021) *TCP port scanner, spews SYN packets asynchronously, scanning entire Internet in under 5 minutes.* Available at: https://github.com/robertdavidgraham/masscan (Accessed: 09th January 2021).

Matherly, J. (2014) 'Inside the world's most dangerous search engine', ShowMe-Con.

Matherly, J. (2017) 'The Complete Guide to Shodan', Austin, leanpub.

Maxim, L. (2016) *Divide and Conquer: High Scalability With MongoDB Sharding.* Available at: https://dzone.com/articles/divide-and-conquer-high-scalability-with-mongodb-t (Accessed: 22th March 2021).

McMillan, R. (2013) *Is It Wrong to Use Data From the World's First 'Nice' Botnet?.* Available at: https://www.wired.com/2013/05/internet-census/ (Accessed: 27th January 2021).

MongoDB. (2021) *MongoDB Limits and Thresholds*. Available at: https://docs.mongodb.com/manual/reference/limits/ (Accessed: 10th March 2021).

Morgan, S. (2019) *The 2019 Official Annual Cybercrime Report - Herjavec Group*. Available at: https://www.herjavecgroup.com/wp-content/uploads/2018/12/CV-HG-2019-Official-Annual-Cybercrime-Report.pdf (Accessed 1st October 2020).

Morisson, B., Riley, C. and Poupino, J. (2016) *INTERNET SECURITY EXPOSURE 2016*. Available at: https://d1ehrggk1349y0.cloudfront.net/BinaryEdge-WorldReport.pdf (Accessed: 17th February 2021).

Morris, A. (2020) *GreyNoise Intelligence Alpha API*. Available at: https://github.com/GreyNoise-Intelligence/api.greynoise.io (Accessed: 18th February 2021).

Motta, J. (2020) *The Secret Coalition Master Plan, or why we acquired BinaryEdge*. Available at: https://www.coalitioninc.com/blog/why-we-acquired-binaryedge (Accessed: 12th February 2021).

Nabha, R. and Sbeyti, H. (2020) 'Exploiting Vulnerabilities Of MRI Scanner Machine: Lebanon Case Study', *2020 8th International Symposium on Digital Forensics and Security (ISDFS).* pp. 1-7. doi: 10.1109/ISDFS49300.2020.9116449.

Nespoli, P., Papamartzivanos, D., Mármol, F. and Kambourakis. (2017) 'Optimal Countermeasures Selection Against Cyber Attacks: A Comprehensive Survey on Reaction Frameworks', *IEEE Communications Surveys & Tutorials,* 20(2), pp. 1361-1396. doi: 10.1109/COMST.2017.2781126.

Nessus (2021) *Nessus Professional.* Available at: https://www.tenable.com/products/nessus/nessus-professional (Accessed: 21st February 2021).

NIST (2018) *National Vulnerability Database (NVD)*. Available at: https://www.nist.gov/programs-projects/national-vulnerability-database-nvd (Accessed: 22th February 2021).

NIST (2020) *Security Content Automation Protocol.* Available at: https://csrc.nist.gov/projects/security-content-automation-protocol (Accessed: 28th February 2021).

NIST (2020a) *CVE-2018-16843 Detail.* Available at: https://nvd.nist.gov/vuln/detail/CVE-2018-16843 (Accessed: 28th February 2021).

NIST (2021) *Official Common Platform Enumeration (CPE) Dictionary.* Available at: https://nvd.nist.gov/products/cpe (Accessed: 24th February 2021).

Nordine, J. (2021) *OSINT Framework.* Available at: https://osintframework.com/ (Accessed: 25th February 2021).

Nouh, M., Nurse, J., Webb, H. and Goldsmith, M. (2019) 'Cybercrime investigators are users too! Understanding the socio-technical challenges faced by law enforcement', *2019 Workshop Usable Security.*

O'Hare, J., Macfarlane, R. and Lo, O. (2019) 'Identifying Vulnerabilities Using Internet-Wide Scanning Data', *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)*, pp. 1-10. doi: 10.1109/ICGS3.2019.8688018.

Onyphe (2019) On Twitter. Available at: https://twitter.com/onyphe/status/1167160300883316739 (Accessed: 11th February 2020).

Onyphe (2020) *ONYPHE - Cyber Defense Search Engine.* Available at: https://www.onyphe.io/ (Accessed: 1st December 2020).

Onyphe (2020a) *ONYPHE – pricing.* Available at: https://www.onyphe.io/pricing/ (Accessed: 1st December 2020).

OpenVAS (2020) *OpenVAS - OpenVAS - Open Vulnerability Assessment Scanner.* Available at: https://www.openvas.org/ (Accessed: 21st February 2021).

Panetta, K. (2016) *Gartner's Top 10 Security Predictions 2016.* Available at: https://www.gartner.com/smarterwithgartner/top-10-security-predictions-2016 (Accessed: 12th February 2021).

Pastor-Galindo, J., Nespoli, P., Mármol. and Pérez, M. (2020) 'The Not Yet Exploited Goldmine of OSINT: Opportunities, Open Challenges and Future Trends', *IEEE Access,* 8, pp. 10282-10304. doi: 10.1109/ACCESS.2020.2965257.

Patton, M., Gross, E., Chinn, R., Forbis, S., Walker, L. and Chen, H. (2014) 'Uninvited Connections: A Study of Vulnerable Devices on the Internet of Things (IoT)' *2014 IEEE Joint Intelligence and Security Informatics Conference*, pp. 232-235. doi: 10.1109/JISIC.2014.43.

Pinterest (2021) *UI – Billing Systems.* Available at: https://www.pinterest.co.uk/weidner_art/ui-billing-systems/ (Accessed: 21st March 2021).

Pritchard, S. (2020) *Shodan founder John Matherly on IoT security, dual-purpose hacking tools, and information overload.* Available at: https://portswigger.net/daily-swig/shodan-founder-john-matherly-on-iot-security-dual-purpose-hacking-tools-and-information-overload (Accessed: 08th January 2021).

Rapid7 (2021) *Reverse DNS (RDNS)*. Available at: https://opendata.rapid7.com/sonar.rdns_v2/ (Accessed: 3nd October 2020).

Rashid, F. (2018*) MAPPING THE INTERNET, WHO'S WHO? (PART THREE).* Available at: https://duo.com/decipher/mapping-the-internet-whos-who-part-three (Accessed: 17th February 2021).

Reitz, K. and Schlusser, T., n.d. The hitchhiker's guide to Python.

Rodrigues, P. (2019) *An OSINT Approach to Automated Asset Discovery and Monitoring.* Available at: https://repositorio-aberto.up.pt/bitstream/10216/119207/2/318708.pdf (Accessed: 2nd October 2020).

Rodriguez, L., Trazzi, J., Fossaluza,V., Campiolo,R., Batist, D. (2018) 'Analysis of Vulnerability Disclosure Delays from the National Vulnerability Database'.

Salame, W. (2020) *Zoomeye hacker search engine*. Available at: https://kalitut.com/zoomeye-search-engine/ (Accessed: 29th January 2021).

Sanguino, L. and Uetz, R. (2017) 'Software Vulnerability Analysis Using CPE and CVE' *ArXiv.*

Schearer, M. (2010) Shodan for Penetration Testers. DEFCON. Las Vegas: DEFCON.

Schlette D., Menges F., Baumer T. and Pernul G. (2020) Security Enumerations for Cyber-Physical Systems, *Data and Applications Security and Privacy XXXIV*, 12122, pp 64-76. doi: 10.1007/978-3-030-49669-2_4.

Shere, A. (2020) 'Now you [don't] see me: how have new legislation and changing public awareness of the UK surveillance state impacted OSINT investigations?', *Journal of Cyber Policy,* 5(3), pp 429-448. doi: 10.1080/23738871.2020.1832129.

Sherry, C. (2020) *Advantages and Disadvantages of Active vs. Passive Scanning in IT and OT Environments*. Available at: https://www.infosecurity-magazine.com/opinions/active-passive-scanning/ (Accessed: 13th January 2021).

Shodan (2020) *Shodan Credits Explained*. Available at: https://help.shodan.io/the-basics/credit-types-explained (Accessed: 15th February 2020).

Shodan (2021) *Shodan*. Available at: https://www.shodan.io/ (Accessed: 1st December 2020).

Shodan (2021a) *Shodan Enterprise - Compare Products.* Available at: https://enterprise.shodan.io/product-comparison (Accessed: 17th February 2021).

Shodan (2021b) *Shodan 2000*. Available at: https://2000.shodan.io/ (Accessed: 17th February 2021).

Shodan (2021c) *Shodan ICS Radar.* Available at: https://ics-radar.shodan.io/ (Accessed: 17th February 2021).

Shodan (2021d) *Shodan Developer.* Available at: https://developer.shodan.io/api (Accessed: 17th February 2021).

Shori, A. (2018) *To Block or not to Block? Impact and Analysis of Actively Blocking Shodan Scans. Available* at: https://www.sans.org/reading-room/whitepapers/networksecurity/block-block-impact-analysis-actively-blocking-shodan-scans-38645 (Accessed: 8th January 2021).

Simon, K., Moucha, C. and Keller, J. (2017) 'Contactless Vulnerability Analysis using Google and Shodan', *Journal of Universal Computer Science*, 23(4), pp. 404-430. doi: 10.3217/jucs-023-04-0404.

Singh, A. (2019) *Graph Analytics and Big Data*. Available at: https://datascience.foundation/sciencewhitepaper/graph-analytics-and-big-data (Accessed: 17th April 2021).

Söderström, U., Carlsson, L. and Soderstrom, U (2019) 'Comparing Millennials View on Minimalism And Maximalism in Web Design', *Proceedings of the 31st European Conference on Cognitive Ergonomics*, pp. 92–95. doi: 10.1145/3335082.3335104.

SpaCy (2021) *SpaCy - Industrial-strength Natural Language Processing in Python* .Available at: https://spacy.io/ (Accessed: 18th February 2021).

Spiderfoot (2021) The OSINT Platform for Attack Surface Monitoring. Available at: https://www.spiderfoot.net/ (Accessed: 22nd February 2021).

Synopsys (2020) *2020 OPEN SOURCE SECURITY AND RISK ANALYSIS REPORT.* Available at: https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/2020-ossra-report.pdf (Accessed: 22th February 2021).

Tanaka, H. (2019) *editdistance - Fast implementation of the edit distance(Levenshtein distance)*. Available at: https://github.com/roy-ht/editdistance (Accessed: 10th April 2021).

Team-Cymru (2021) *IP to ASN Mapping Service*. Available at: https://team-cymru.com/community-services/ip-asn-mapping/(Accessed: 10th March 2021).

Tripwire IP360 (2019) *TRIPWIRE IP360 DATASHEET.* Available at: https://www.tripwire.com/-/media/tripwiredotcom/files/datasheet/tripwire_ip360_product_datasheet.pdf?rev=390bcffdc54247c48526705115545e1a (Accessed: 21st February 2021).

Twitter (2021) *Onyphe*. Available at: https://twitter.com/onyphe/status/1355812330894983168 (Accessed: 14th April 2021).

UKEssays (2018) *Rapid Application Development vs. Agile Methodologies.* Available from: https://www.ukessays.com/essays/management/rapid-application-development-vs-agile-methodologies.php (Accessed: 22nd March 2021).

Uptrends (2021) *Website Monitoring and Web Performance Monitoring | Uptrends*. Available at: https://www.uptrends.com/ (Accessed: 15th March 2021).

Waltermire, D., Quinn, S., Scarfone, K and Prisaca, D. (2018) 'The Technical Specification for the Security Content Automation Protocol (SCAP) Version 1.3', *NIST Special Publication*. doi: NIST.SP.800-126r3.pdf.

Wan, G., Izhikevich, L., Adrian, D., Yoshioka, K., Holz, R., Rossow, C. and Durumeric, Z. (2020) 'On the Origin of Scanning: The Impact of Location on Internet-Wide Scans', *IMC '20: Proceedings of the ACM Internet Measurement Conference*, 20(1), pp. 662-679. DOI: 3419394.3424214.

Wåreus, E. and Hell, M. (2020) 'Automated CPE Labeling of CVE Summaries with Machine Learning', *Detection of Intrusions and Malware, and Vulnerability Assessment, 122223*(1), pp. 3-22. DOI: 978-3-030-52683-2_1.

Weimer, F. (2005) 'Passive DNS Replication', *In Proceedings of 17th Annual FIRST Conference on Computer Security Incident Handling.*

William, J. (2020) *Identification of IP addresses using fraudulent geolocation data.* B.E. Individual Project. Imperial College London. Available at: https://www.imperial.ac.uk/media/imperial-college/faculty-of-engineering/computing/public/1920-ug-projects/Williams,-James-(jw1317).pdf (Accessed: 18 March 2021).

Wright, T. (2020) *With a backdrop of increased cybercriminal activity due to COVID-19, the UK is taking steps to regulate the security of internet-connected devices.* Available at: https://www.fladgate.com/2020/05/with-a-backdrop-of-increased-cybercriminal-activity-due-to-covid-19-the-uk-is-taking-steps-to-regulate-the-security-of-internet-connected-devices/ (Accessed: 12th January 2021).

Xu, X., Wan, J., Zhang, W., Tong, C. and Wu, C. (2011) 'PMSW: a passive monitoring system in wireless sensor networks', *International Journal of Network Management*, 21(4), pp. 300-325. doi: 10.1002/nem.792.

Xuanzhen, G., Zulie, P., Yuanchao, C. (2020) 'Application of Passive DNS in Cyber Security', *2020 IEEE International Conference on Power, Intelligent Computing and Systems*, pp. 257-259. doi: 10.1109/ICPICS50287.2020.9202344.

xxdesmus. (2020) *Thai Database Leaks 8.3 Billion Internet Records.* Available at: https://rainbowtabl.es/2020/05/25/thai-database-leaks-internet-records/ (Accessed: 28th January 2021).

Zalewski, M. (2014) *p0f v3.* Available at: https://lcamtuf.coredump.cx/p0f3/ (Accessed: 21st February 2021).

Zdrnja, B., Brownless, N. and Wessels, D. (2007) 'Passive Monitoring of DNS Anomalies', *Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 129-139.* doi: 10.1007/978-3-540-73614-1_8.

Zhao, B., Ji, S., Lee, W., Lin, C., Weng, H., Wu, J., Zhou, P., Fang, L. and Beyah, R. (2020) 'A Large-scale Empirical Study on the Vulnerability of Deployed IoT Devices', *IEEE Transactions on Dependable and Secure Computing*. doi: 10.1109/TDSC.2020.3037908.

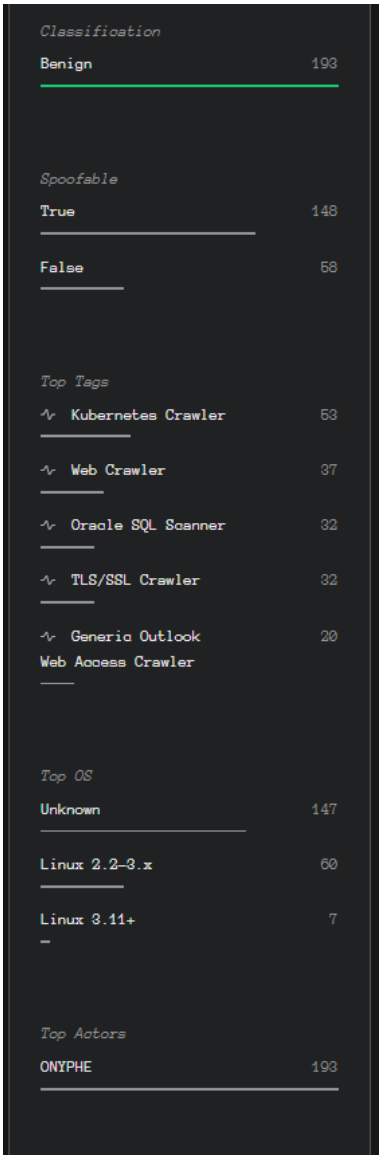
ZMap (2020) *The ZMap Project*. Available at: https://zmap.io/ (Accessed: 29th December 2020).

ZoomEye (2021) *ZoomeEye – Cyberspace Engine.* Available at: https://www.zoomeye.org/ (Accessed: 08th February 2021).

APPENDICES

APPENDIX A – ONYPHE GREYNOISE RESULTS

Onyphe GreyNoise Results

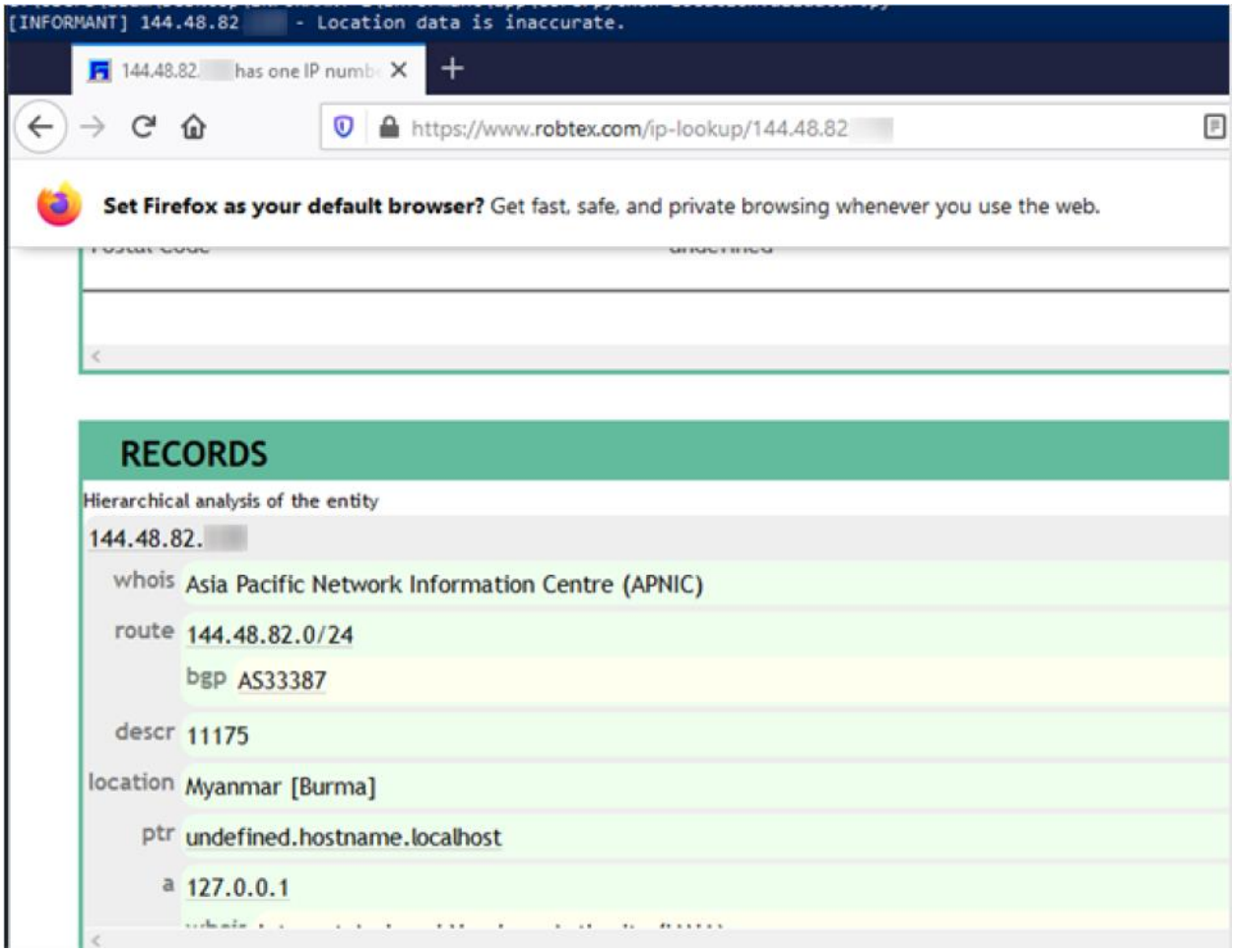


APPENDIX B – FRAUDULENT GEOLOCATION VALIDATION

Fraudulent Geolocation Validation Code

```
import requests, platform, subprocess

def location_distance(origin, dest):
    """
    Function to calculate distance between orgin and host
    """
    try:
        resp = requests.get(url="https://www.distance24.org/route.json?stops=" + origin + "|" + dest)
        data = resp.json()
        distFromOrigin = 0.621371 * data['distance']
        return distFromOrigin
        if (resp.status_code == 404):
            print(f"Distance not found.")
    except:
            print(f"Distance24 connection error occured.")
            return 0

def ping_check(host):
    """
    Returns avg ping time value if host (str) responds to a ping request.
    """

    # Option for the number of packets as a function of
    param = '-n' if platform.system().lower()=='windows' else '-c'

    # Building the command. Ex: "ping -c 1 google.com"
    command = ['ping', param, '1', host]
    #command = 'ping ' + param + ' 1 ' + host
    try:
        print(command)
        response = subprocess.check_output(command, stdout=subprocess.PIPE)
        print(response)
        avgPing = str(response, 'utf-8')

        if "Average = " in avgPing:
            index = avgPing.find("Average") + 10
            avgPing = avgPing[index:-4]
            return avgPing
    except:
        print("Location validation error: Please ensure the correct privileges are allowed.")
        return 0


def location_verification(distFromOrigin, avgPingTime):

    c = 124188.0
    calc = (distFromOrigin / c) * 1000

    if (calc > float(avgPingTime)):
        return False
    else:
        return True


def validate_loc(orign, dest, host):
    distFromOrigin = location_distance(orign , dest)
    if distFromOrigin > 0:
        avgPingTime = ping_check(host)

    # Just return funcation below after testing
    if location_verification(distFromOrigin, avgPingTime):
        print ("[INFORMANT] " + str(host) + " - Location data is accurate.")
        return True
    else:
        print ("[INFORMANT] " + str(host) + " - Location data is inaccurate.")
        return False
```

Fraudulent Geolocation Validation Unit Test Proof



APPENDIX C – CPE CONSTRUCTION ALGORITHM

Core CPE Construction Algorithm

```
def banner_to_cpe(manufacturer, vendor, product, version, file_mode, cpe_list):
    jStr = ':'
    print("Original service: " + manufacturer + " " + vendor + " " +  product + " " + version)
    cpeArray = ['cpe:2.3', 'a', manufacturer, product, version, '*', '*', '*', '*', '*', '*', '*']
    p = jStr.join(cpeArray).lower()
    print("Original CPE: " + p)
    cpeString = exceptions(cpeArray[2].lower(), vendor.lower() ,cpeArray[3].lower(), cpeArray[4].lower())

    if file_mode:
        with open('cpedb.txt', 'r') as f:
            data = [line.strip() for line in f]
    else:
        data = cpe_list

    print("[INFORMANT] Verifying CPE Value")

    if cpeString != "cpe:2.3:a:::*:*:*:*:*:*:*:*":
        cpe = process.extractOne(cpeString, data, score_cutoff = 90)
        print("Generated CPE (after exceptions): " + cpeString)

    try:
        print("Predicted CPE: " + str(cpe[0]))
        print("Match Probability: " + str(cpe[1]) + "%")
        print("Similarity: " + str(round(fuzz.ratio(cpe[0], cpeString), 2)) + "%")
        return(cpe[0], cpe[1])
    except Exception:
        print("Generated CPE could not be verified. Using generated CPE value.")
        return(cpeString, 'No Match')
```

APPENDIX D – CPE EXCEPTIONS CODE

CPE Exception Code

```python
def exceptions(manufacturer, vendor, product, version):
    jStr = ':'
    cpeArray = ['cpe:2.3', 'a', manufacturer, product, version, '*','*', '*', '*', '*', '*', '*']

    """ SSH CHECKS """

    if vendor == 'openssh' or manufacturer == 'openssh' or product == 'openssh':
        pl = '*'
        # Check for package in version number
        if any(c.isalpha() for c in version):
            pl = (version[version.find('p'):])
            version = (version[:version.find('p')])

            # Further check to get rid of Ubuntu junk from pl (BinaryEdge results)
            if 'ubuntu' in pl:
                pl=pl[:2]

        cpeArray[2] = 'openbsd'
        cpeArray[3] = 'openssh'
        cpeArray[4] = version
        cpeArray[5] = pl

    """ WEB SERVER CHECKS """
    if ('nginx' in vendor or 'nginx' in manufacturer or 'nginx' in product):
        cpeArray[2] = 'nginx'
        cpeArray[3] = 'nginx'

    if (('microsoft' in vendor or 'microsoft' in manufacturer or 'microsoft' in product) and ('iis' in vendor or 'iis' in manufacturer or 'iis' in product)):
        cpeArray[2] = 'microsoft'
        cpeArray[3] = 'internet_information_server'
    if (('apache' in vendor or 'apache' in manufacturer or 'apache' in product) and ('httpd' in vendor or 'httpd' in manufacturer or 'httpd' in product)):
        cpeArray[2] = 'apache'
        cpeArray[3] = 'http_server'
    if vendor == 'httpapi' or manufacturer == 'httpapi' or product == 'httpapi' or 'httpapi' in product:
        cpeArray[2] = 'wampserver'
        cpeArray[3] = 'wampserver'
    if vendor == 'lighttpd' or manufacturer == 'lighttpd' or product == 'lighttpd':
        cpeArray[2] = 'lighttpd'
        cpeArray[3] = 'lighttpd'
    if vendor == 'sendmail' or manufacturer == 'sendmail' or product == 'sendmail':
        cpeArray[2] = 'sendmail'
        cpeArray[3] = 'sendmail'
    if vendor == 'postfix' or manufacturer == 'postfix' or product == 'postfix':
        cpeArray[2] = 'postfix'
        cpeArray[3] = 'postfix'
    cpeString = jStr.join(cpeArray).lower()

    return cpeString
```

APPENDIX E – CVE LOOKUP AND ASSIGNMENT

Valid CPE List Retrieval

```python
def gen_local_cpe_file(option, project_name):

    data = []
    master_list = []

    try:
        data = list(mainDB[project_name].find({"source": 'merged'}).sort("timestamp", -1))
    except Exception as e:
        print("Failed to obtain asset list from DB: " + str(e))
        return master_list, data

    if option:
        if (os.path.isfile("cpedb.txt")) :
            print("[INFORMANT] Skipping local DB file created.")
        else:
            try:
                cpeList = vulDB['cpe'].find({})
                for record in cpeList:
                    if 'cpe_2_2' in record:
                        master_list.append(record['cpe_2_2'])
                    if 'cpe_name' in record:
                        for entry in record['cpe_name']:
                            try:
                                master_list.append(entry['cpe23Uri'])
                            except Exception:
                                continue

                f = open("cpedb.txt", "w")
                for cpe in master_list:
                    f.write(cpe + '\n')
                f.close()
                print("[INFORMANT] Created local CPE list from DB")
            except Exception as e:
                print("Failed to obtain cpe list from DB: " + str(e))

        return None, data
    else:
        try:
            cpeList = vulDB['cpe'].find({})
            for record in cpeList:
                if 'cpe_2_2' in record:
                    master_list.append(record['cpe_2_2'])
                if 'cpe_name' in record:
                    for entry in record['cpe_name']:
                        try:
                            master_list.append(entry['cpe23Uri'])
                        except Exception:
                            continue
        except Exception as e:
            print("Failed to obtain cpe list from DB: " + str(e))

        return master_list, data
```

CVE Lookup

```
def cve_lookup(cpe, version, service, score) :
    # Find known vulnerabilties matching vulnerable config
    cves = vulDB['cves'].find({"vulnerable_configuration" : cpe})
    cveList = []

    if (cves.count() > 0 and version != '') :
        print ("Vulnerabilities found:")
        for cve in cves :
            print("CVE: " + str(cve["id"]))
            cveFound = {'cve' : cve["id"], 'cvss2': cve["cvss"], 'vector' : cve["cvss-vector"], 'cpe' : cpe, "port": '', "manufacturer": '', "product": '', "version": '', "cpe_score": '', 'vul_by_default': False}
            cveFound['port'] = service.get('port', '')
            cveFound['manufacturer'] = service.get('manufacturer', '')
            cveFound['product'] = service.get('product', '')
            cveFound['version'] = service.get('version', '')
            cveFound['vul_by_default'] = default_check(cveFound)
            try:
                cveFound['cpe_score'] = score
            except:
                cveFound['cpe_score'] = 'Not found'

            cveList.append(cveFound)
            if (cve["cvss-vector"][:4] == "AV:N") :
                print("CVSS2: " + str(cve["cvss"]) + "                          Remotely exploitable.")
            elif (cve["cvss-vector"][:4] == "AV:L"):
                print("CVSS2: " + str(cve["cvss"]) + "                          Locally exploitable.")
            elif (cve["cvss-vector"][:4] == "AV:A"):
                print("CVSS2: " + str(cve["cvss"]) + "                          Exploitable via Adjcacent Network.")
            elif (cve["cvss-vector"][:4] == "AV:P"):
                print("CVSS2: " + str(cve["cvss"]) + "                          Physically exploitable.")
            else :
                print("CVSS2: " + str(cve["cvss"]))
        return cveList
    elif (version == '') :
        return 'obfuscated'
    else:
        print ("No vulnerabilities found.")
        print("--------------------------------------------------------------------------")
        return None
    print("--------------------------------------------------------------------------")
```

CVE Assignment

```
def vul_scan(project_name):

    # Declare required vulnerables
    project_data = list(mainDB[project_name].find({'project_name' : project_name}))
    file_mode = True
    bulk_updates = []

    # Save histroical stats
    save_last_project_stats(project_name)

    # Generate local CPE file (.txt) - True = generate local .txt file | False = hold data in memory (Faster but more resource intensive)
    cpe_list, asset_list = gen_local_cpe_file(file_mode, project_name)

    # Validation/Error checking
    if (len(asset_list) == 0):
        print("[INFORMANT] No assets to check, run an Asset scan first.")
        exit(1)
    elif cpe_list is None:
        print("[INFORMANT] Using file-based fuzzer.")
    elif (len(cpe_list) >= 1):
        print("[INFORMANT] Using memory-based fuzzer.")
    else:
        print("[INFORMANT] No CPE data available, check the state of the CVE DB.")
        exit(1)

    validAssets = latest_records(asset_list, project_data[0])

    for asset in validAssets:
        try:
            mainDB[project_name].update({"_id": asset['_id']},{"$unset": {"cveData": 1}}, upsert=False)
            mainDB[project_name].update({"_id": asset['_id']},{"$unset": {"risks": 1}}, upsert=False)

            for service in asset['banners']:
                cpe_string, score = banner_to_cpe(service.get('manufacturer', ''),service.get('vendor', ''),service.get('product', ''),service.get('version', ''), file_mode, cpe_list)
                cves = cve_lookup(cpe_string, service.get('version', ''), service, score)
                if (cves is not None and cves != 'obfuscated'):
                    for cve in cves:
                        if default_check(cve):
                            cur_service = {'service' : service, 'cve' : cve, 'default' : True}
                            bulk_updates.append(UpdateOne({'_id': asset['_id']}, {'$push': {'risks': cur_service}}))
                    bulk_updates.append(UpdateOne({'_id': asset['_id']}, {'$push': {'cveData': cves}}))
                elif (cves == 'obfuscated'):
                    cur_service = {'service' : service, 'obfuscated' : True}
                    bulk_updates.append(UpdateOne({'_id': asset['_id']}, {'$push': {'risks': cur_service}}))
                else:
                    continue

        except Exception as e:
            print(f'[INFORMANT] Fatal CVE assignment error: {e}')

    # Update MongoDB in batches
    try:
        mainDB[project_name].bulk_write(bulk_updates)
    except Exception as e:
        print(str(e))

    # Need to rerun port flag and rddos check
    port_flag_check(project_name)
    rddos_prediction(project_name)

    # Update last_run time
    save_project_stats(project_name)
```

APPENDIX F – PASSIVE DNS ENRICHMENT CODE SNIPPET

Passive DNS Enrichment Code Snippet

```
    if len(pdnsIpList) >= 50:
        print("[INFORMANT] Using Bulk ASN & Netname Lookup - Total IPs: " + str(len(pdnsIpList)))
        # Batch up the bulk requests
        batches = [pdnsIpList[x:x+500] for x in range(0, len(pdnsIpList), 500)]
        if(len(batches) > 1):
            print("[INFORMANT] Large Data Set Detected! Total Batches: " + str(len(batches)))
            delay = 1

        for batch in batches:
            i += 1
            bulk_query = "begin\nverbose\n"
            for ip in batch:
                bulk_query += str(ip) + '\n'
            bulk_query += "end"
            time.sleep(delay)
            response = cymru_query(bulk_query.encode('utf_8'), 5)
            string = response.decode('utf-8')
            bulk_updates = []
            # Quick RE clean up (probs better changing into CSV format first)
            for entry in string.splitlines():
                try:
                    split_entry = entry.split("|", 6)
                    asn = split_entry[0].strip()
                    ip = split_entry[1].strip()
                    netname = split_entry[6].strip()
                    netname = netname.split(",", 1)
                    bulk_updates.append(UpdateMany({"$and": [{"ip": ip}, {'timestamp': timestamp}]}, {'$set': {'asn': asn}}))
                    bulk_updates.append(UpdateMany({"$and": [{"ip": ip}, {'timestamp': timestamp}]}, {'$set': {'netname': netname[0]}}))
                except Exception as e:
                    print(str(e))
                    continue
            try:
                mainDB[project_name].bulk_write(bulk_updates)
            except BulkWriteError as bwe:
                pprint(bwe.details)
            print("[INFORMANT] Batch " + str(i) + " of " + str(len(batches)) + " complete")

    else:
        print("[INFORMANT] Using Standard ASN & Netname Lookup")
        for record in mainDB[project_name].find({'timestamp': timestamp}):
            if 'type' in record:
                if (not record['ip'].islower() and not record['ip'].isupper()):
                    try:
                        lookup = IPWhois(record['ip']).lookup_rdap(asn_methods=['dns', 'whois', 'http'])
                        mainDB[project_name].find_one_and_update({"_id": record['_id']},{"$set": {"asn": lookup['asn']}})
                        mainDB[project_name].find_one_and_update({"_id": record['_id']},{"$set": {"netname": lookup['network']['name']}})
                    except Exception as e:
                        print("[INFORMANT] PDNS ASN & Netname Enrichment Error (Standard method): " + str(e))
                        continue
```

APPENDIX G – DEFAULT CONFIGURATION CVE CHECK

Implemented Vector Approach

```
707  def default_check(cve):
708
709      remote = False
710      default = False
711
712      if (cve["vector"][:4] == "AV:N") :
713          remote = True
714      if (cve["vector"][5:9] == "AC:L"):
715          default = True
716
717      if (remote and default):
718          return True
719      else:
720          return False
```

Implemented Vector Approach Result – Overview Page

Thu, 01 Apr 2021 21:10:00 GMT	127.0.0.1	AS786	test.ac.uk test.com	University of Informant	United Kingdom	Linux Ubuntu	22	OpenSSH 7.6p1 Ubuntu-4ubuntu0.3 (22) **3 issues**	DF (CVE-2018-15473) DF (CVE-2018-15919)

Implemented Vector Approach Result – Asset Drilldown Page

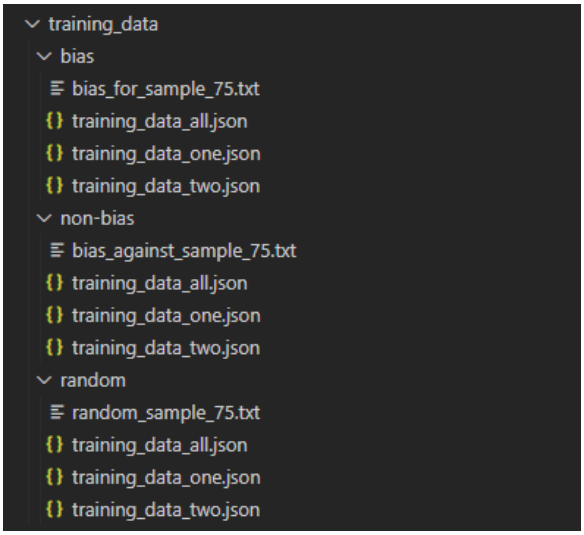
Service	CVE	CVSSv2 Score	Vector	Port	Confidence
OpenSSH 7.6p1 Ubuntu-4ubuntu0.3	DF (CVE-2018-15473)	5	AV:N/AC:L/Au:N/C:P/I:N/A:N	22	100%
OpenSSH 7.6p1 Ubuntu-4ubuntu0.3	DF (CVE-2018-15919)	5	AV:N/AC:L/Au:N/C:P/I:N/A:N	22	100%
OpenSSH 7.6p1 Ubuntu-4ubuntu0.3	CVE-2018-20685	2.6	AV:N/AC:H/Au:N/C:N/I:P/A:N	22	100%

APPENDIX H – PROPOSED MACHINE LEARNING DEFAULT CONFIGURATION CVE CHECK

Proposed Machine Learning Approach Code

```
model_dir = Path("ml/model")

def create_model():
    # load the training data
    with open('ml/TRAINING_DATA.json') as fp:
        training_data = json.load(fp)

    # prepare an empty model to train
    spacy.prefer_gpu()
    nlp = spacy.blank('en') # Blank model
    #nlp.vocab.vectors.name = 'demo'
    ner = nlp.create_pipe('ner')
    nlp.add_pipe(ner, last=True)

    # Add the custome NER Tags as entities into the model
    for label in training_data["classes"]:
        print(label)
        nlp.entity.add_label(label)

    # Train the model
    optimizer = nlp.begin_training()

    for text, annotations in training_data["annotations"]:
        if len(text) > 0: # in case an empty sentence was saved while annotating
            nlp.update([text], [annotations], sgd=optimizer)
            print([text], [annotations])
    # List of pipes you want to train
    pipe_exceptions = ["ner", "trf_wordpiecer", "trf_tok2vec"]

    # List of pipes which should remain unaffected in training
    other_pipes = [pipe for pipe in nlp.pipe_names if pipe not in pipe_exceptions]


    # Begin training by disabling other pipeline components
    with nlp.disable_pipes(*other_pipes) :
        sizes = compounding(1.0, 4.0, 1.001)
        # Training for 30 iterations
        for itn in range(30):
            # shuffle examples before training
            random.shuffle(training_data["annotations"])
            # batch up the examples using spaCy's minibatch
            batches = minibatch(training_data["annotations"], size=sizes)
            # ictionary to store losses
            losses = {}
            for batch in batches:
                texts, annotations = zip(*batch)
                # Calling update() over the iteration
                nlp.update(texts, annotations, sgd=optimizer, drop=0.35, losses=losses)
                print("Losses", losses)

    return nlp

def save_model(nlp):
    if not model_dir.exists():
        model_dir.mkdir()

    nlp.meta['name'] = 'cvedb_model'
    print("Saving model to: ", model_dir)
    nlp.to_disk(model_dir)
    print("Model saved to: ", model_dir)
```

Proposed Machine Learning Approach Training Data Overview



Training Data - Bias for Sample


```
{"classes":["DEFAULT_0", "NON_DEFAULT"],"annotations":[
    ["The default configuration of kdm in Caldera and Mandrake Linux, and possibly other distributions, allows XDMCP
    connections from any host, which allows remote attackers to obtain sensitive information or bypass additional access
    restrictions.",{"entities":[[0,25,"DEFAULT_0"]]}]
```


Training Data - Bias Against Sample


```
{"classes":["DEFAULT_0", "NON_DEFAULT"],"annotations":[
    ["ezconfig.asp in Linksys WRT54G router 3.01.03 , 3.03.6, non-default configurations of 2.04.4, and possibly other
    versions, does not use an authentication initialization function, which allows remote attackers to obtain encrypted
    configuration information and, if the key is known, modify the configuration.",{"entities":[[55,81,"NON_DEFAULT"]]}]
```


Training Data - Random Sample


```
{"classes":["DEFAULT_0", "NON_DEFAULT"],"annotations":[
    ["WebKit in Apple Safari before 5.0 on Mac OS X 10.5 through 10.6 and Windows, and before 4.1 on Mac OS X 10.4, allows
    remote attackers to bypass intended restrictions on outbound connections to \" non-default TCP ports \" via a crafted port
    number, related to an \" integer truncation issue. \" NOTE: this may overlap CVE-2010-1099.",{"entities":[[0,108,
    "DEFAULT_0"]]}]
```

Proposed Machine Learning Approach Model Test Code

```
86  def test_model(text, nlp):
87      print("*******************")
88      print("Input: ", text)
89      doc = nlp(text)
90      if len(doc.ents) > 0:
91          for ent in doc.ents:
92              print("Label: ", ent.label_, "\nReason for Assignment: ", ent)
93              print("*******************")
94      else:
95          print("No labels allocated for: ", doc)
96          print("*******************")
97
```

Proposed Machine Learning Approach Sample Test

```
*****************
Input:  Serv-U FTP server before 5.1.0.0 has a default account and password for local administration, which allows local users to execute arbitrary commands by connect
ing to the server using the default administrator account, creating a new user, logging in as that new user, and then using the SITE EXEC command.
Label:  DEFAULT_0
Reason for Assignment:  has a default account and password for local administration
*****************
```

APPENDIX I – RDDOS PREDICTION

rDDoS Prediction Code

```python
def rddos_prediction(project_name):

    # Update array
    bulk_updates = []

    # Misc info
    project_data = list(mainDB[project_name].find({'project_name' : project_name}))

    # Grab asset data
    data_tmp = list(mainDB[project_name].find({"source" : 'merged'}))
    assets = latest_records(data_tmp, project_data[0])

    # Grab rDDoS data
    with open('app/core/rddos/data.json', 'r') as jsondata:
        rddos_data = json.load(jsondata)

    # Check for ports susceptible to rDDoS
    for asset in assets:
        rddos_prediction = 0
        rddos_port = []
        if 'ports' in asset:
            for rddos in rddos_data['rddos']:
                if isinstance(rddos['port'], int):
                    if rddos['port'] in asset['ports']:
                        rddos_prediction += rddos['baf_max']
                        rddos_port.append(rddos['port'])
                else:
                    for r_port in rddos['port']:
                        if r_port in asset['ports']:
                            rddos_prediction += rddos['baf_max']
                            rddos_port.append(rddos['port'])

        # Save rDDoS BAF if applicable
        if rddos_prediction != 0:
            bulk_updates.append(UpdateOne({'_id': asset['_id']}, {'$push': {'risks': {'rddos' : True, 'rDDoS_BAF' : rddos_prediction, 'port' : rddos_port}}}))

    # Send batched updates
    if len(bulk_updates) > 0:
        try:
            mainDB[project_name].bulk_write(bulk_updates)
        except Exception as e:
            print("[INFORMANT] rDDoS enrichment error: ")
```

rDDoS Prediction Result - Overview Page

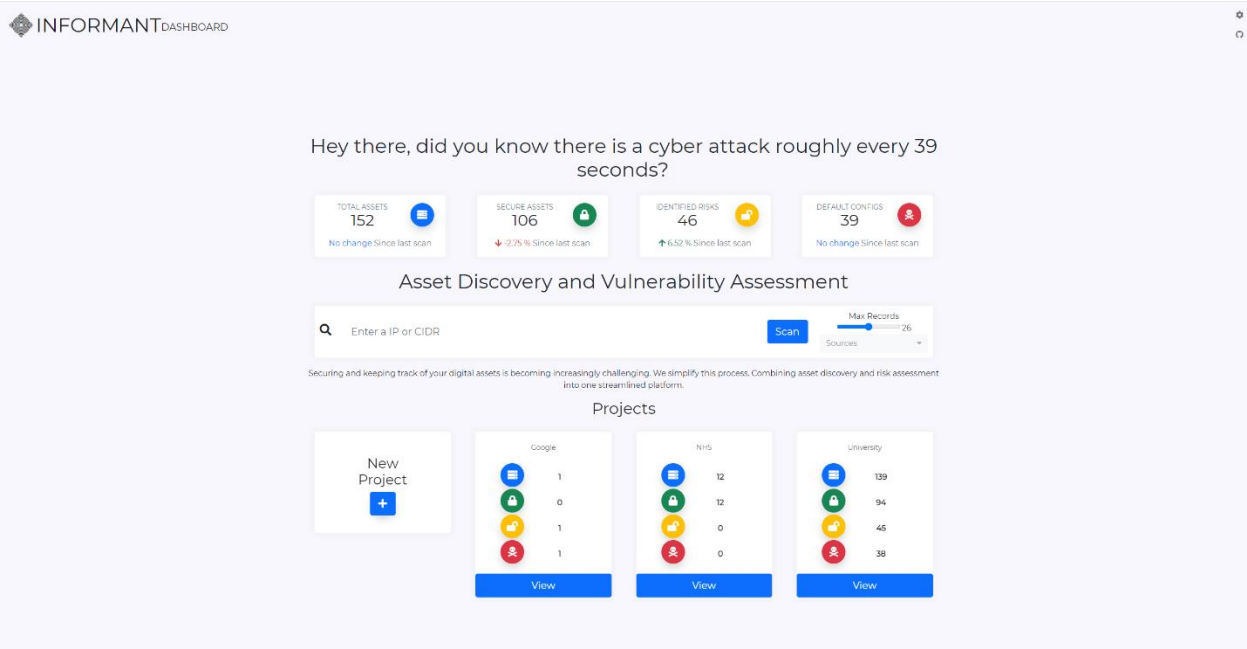
Thu, 01 Apr 2021 21:10:00 GMT	127.0.0.1	786	United Kingdom	unknown	53	None 👍	rDDoS Potential (54)

rDDoS Prediction Result – Asset Drilldown Page

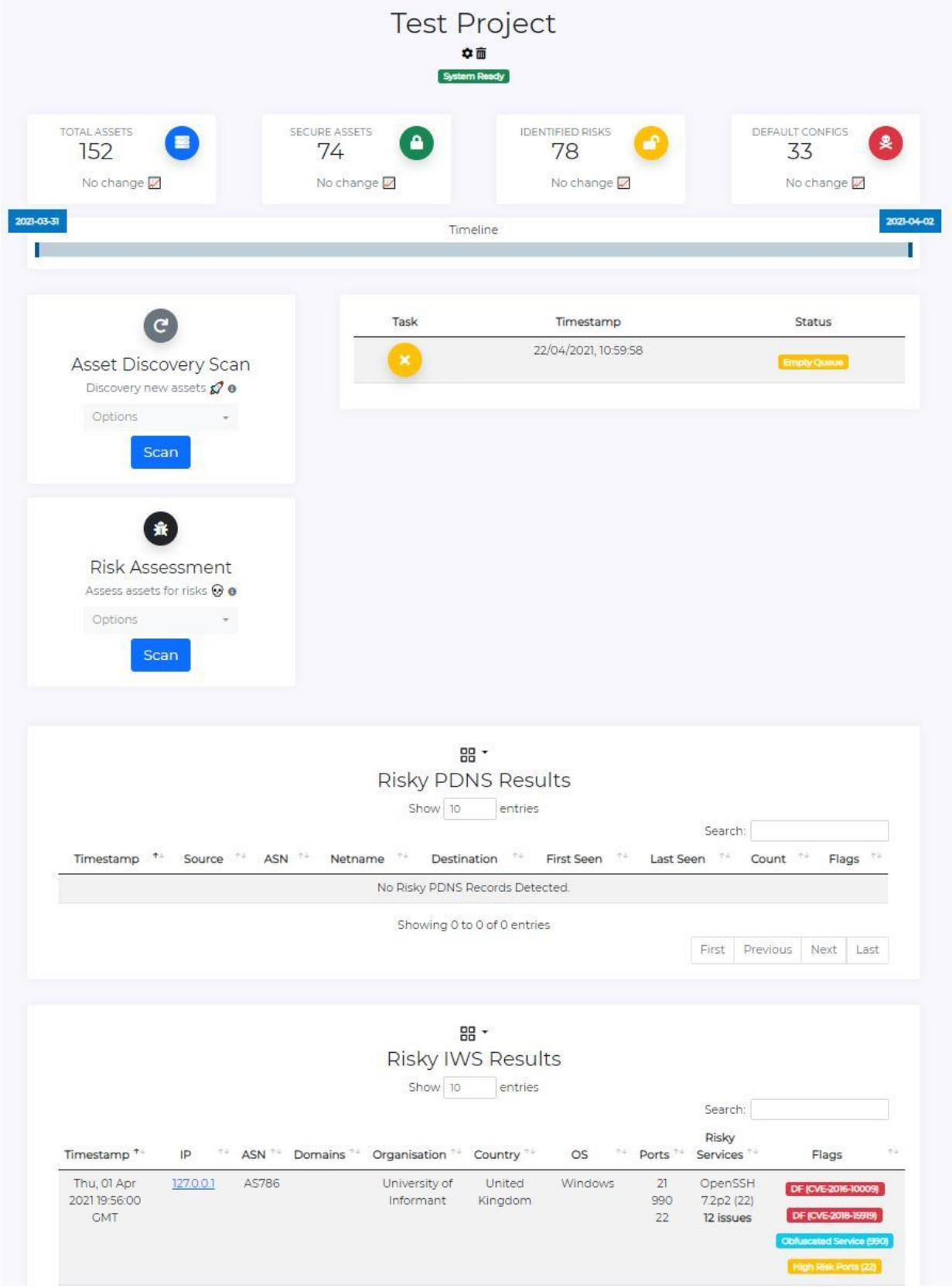
Predicated Total rDDoS (BAF):	54
rDDoS Ports:	53
High Risk Ports:	
Risky Services:	
Default CVEs:	0
Total CVEs:	0
Timestamp:	Thu, 01 Apr 2021 21:10:00 GMT

APPENDIX J – WEB INTERFACE UI

Homepage

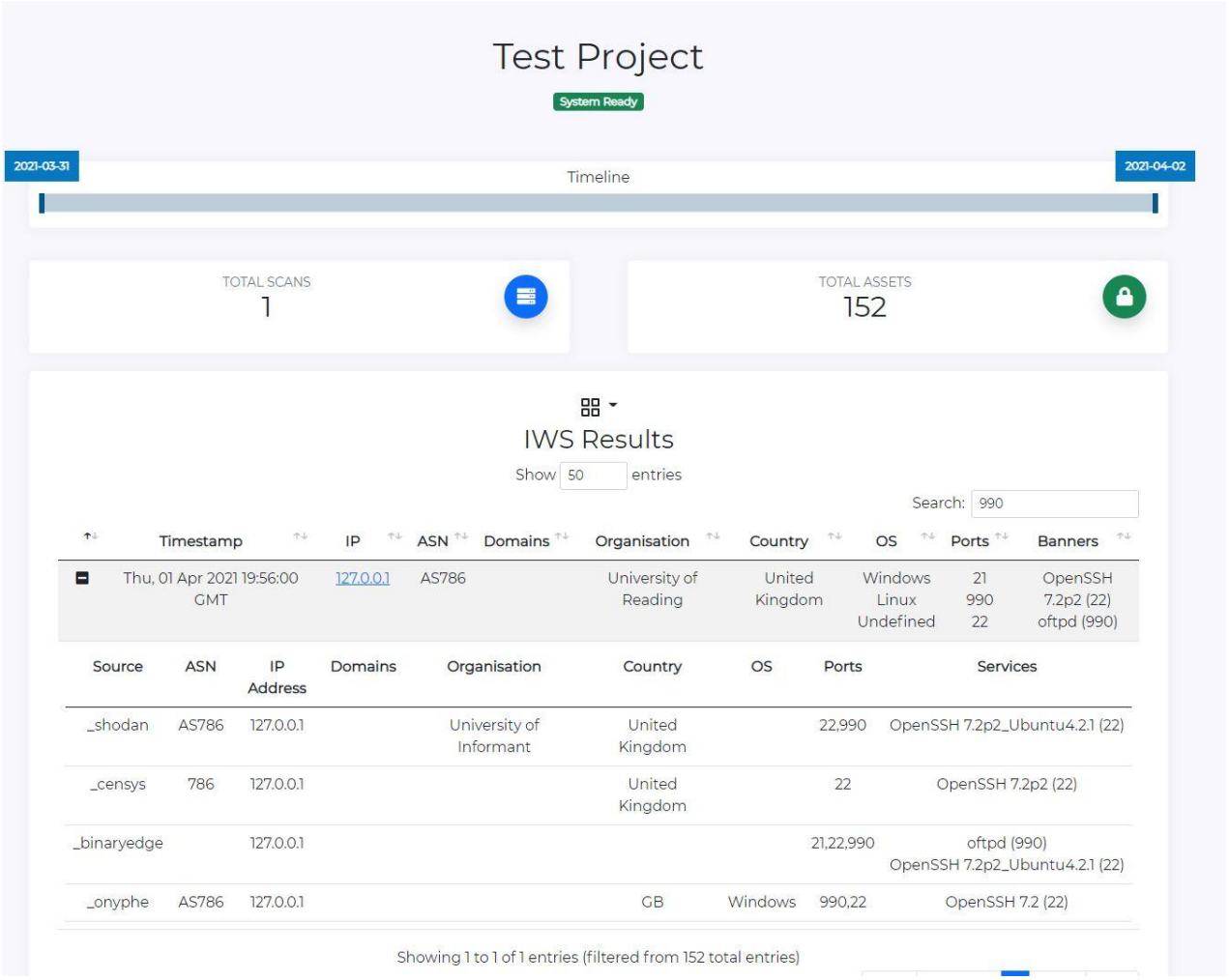


Project Overview Page

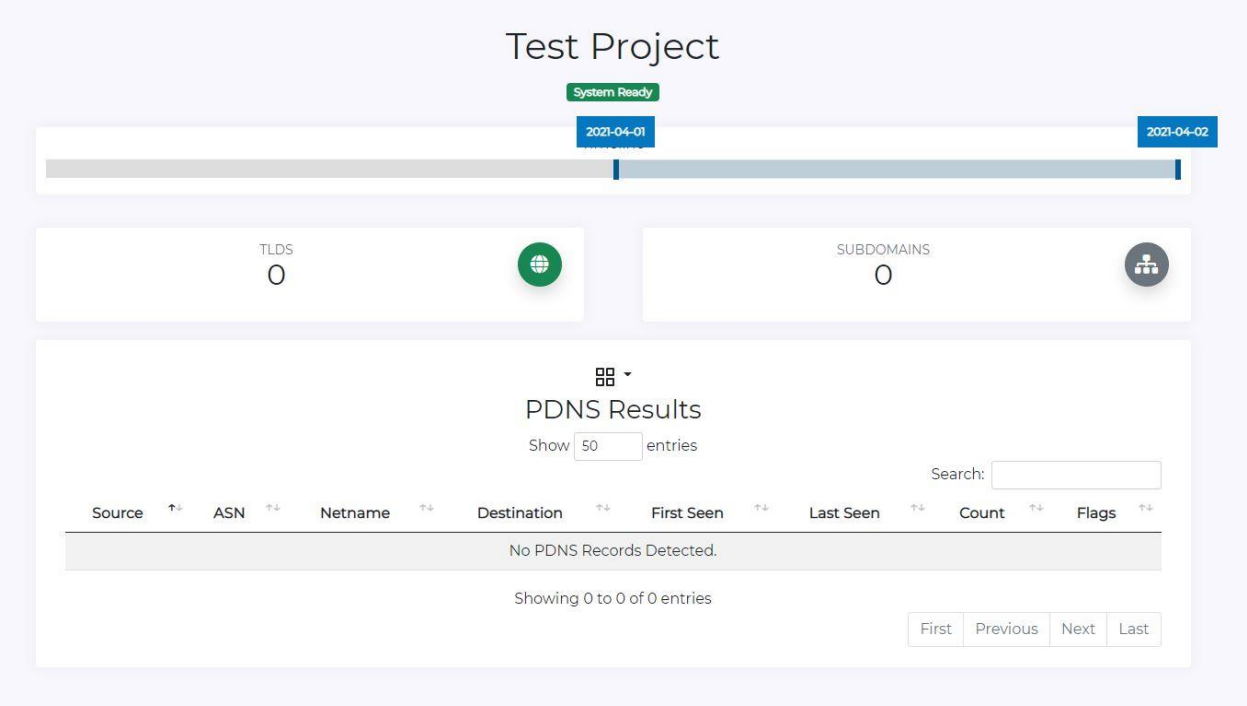


Drilldown Data Dropdown Options

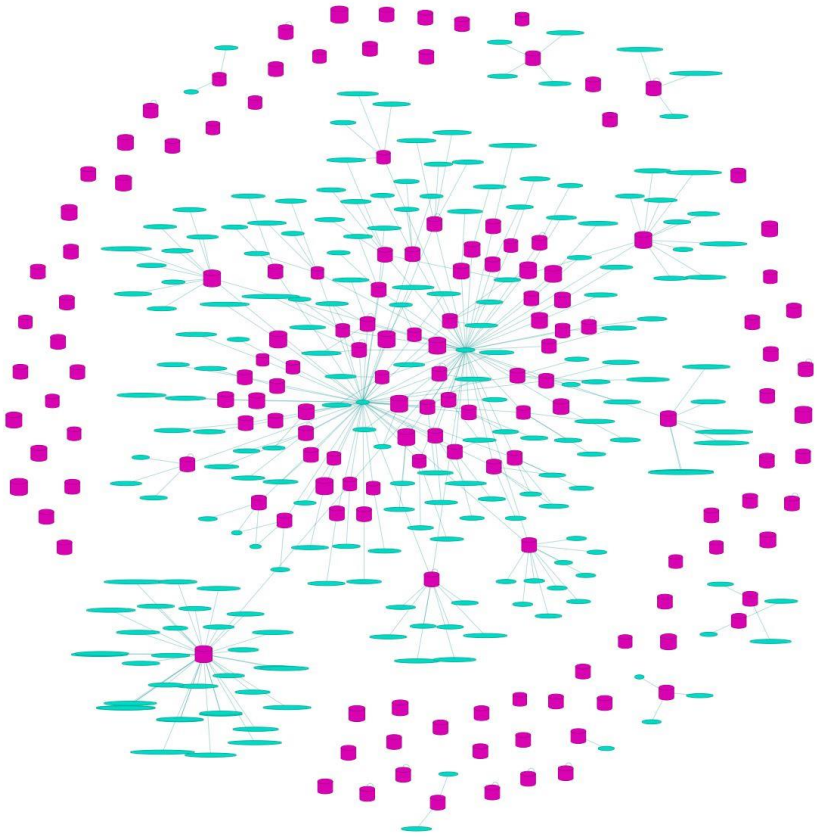
Internet-wide Scanning Breakdown



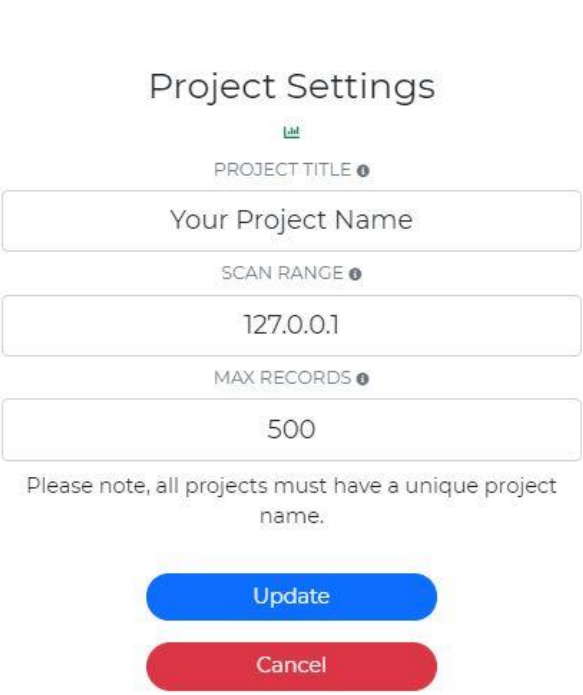
Passive DNS Breakdown



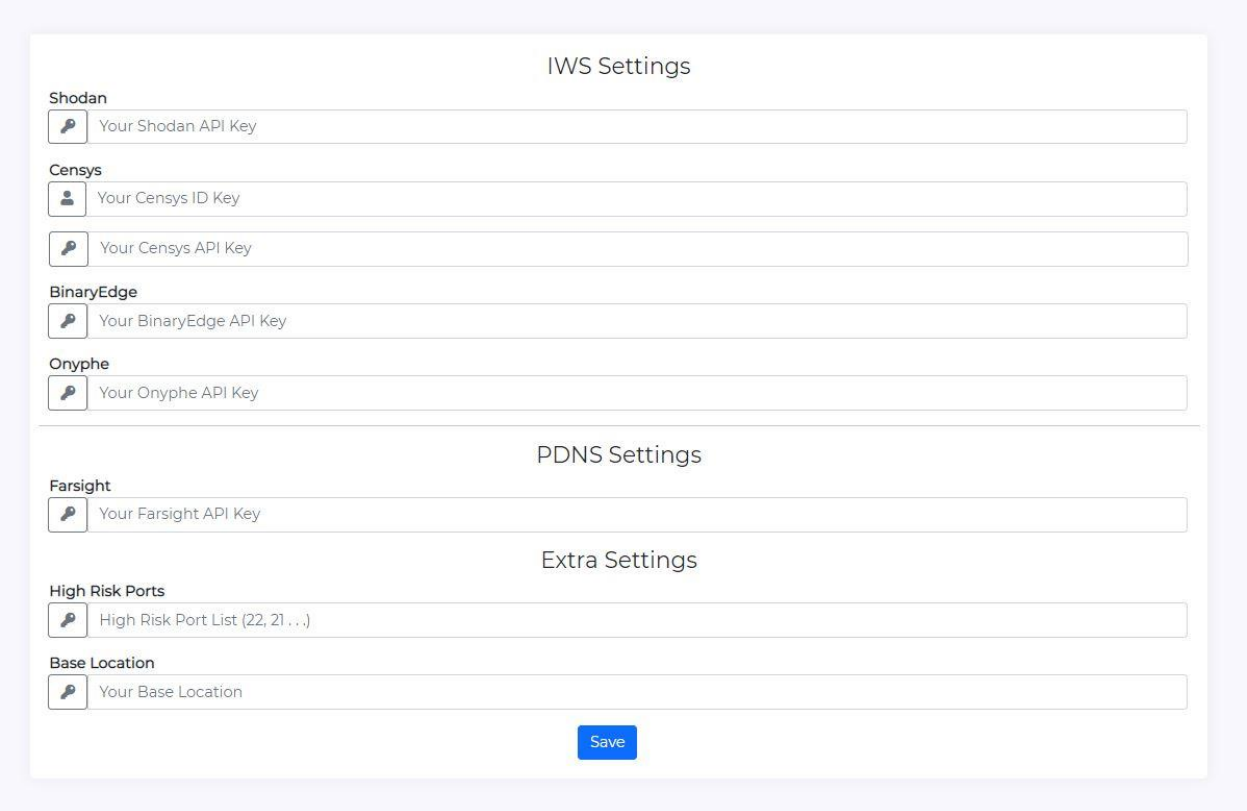
Exposed Attack Surface Map



Project Settings



Core Global System Settings



APPENDIX K – REDIS BASED TASK MANAGEMENT SYSTEM

Example Code for Worker Thread

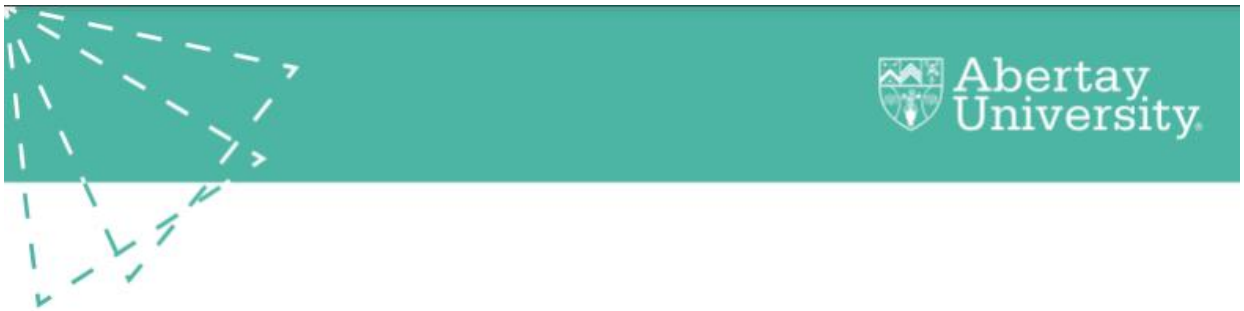
```python
# !/usr/bin/env python
# Name:     worker.py
# By:       LA-Shill
# Date:     22.04.2021
# Version   0.1
# ----------------------------------------------

import os
from dotenv import load_dotenv
import redis
from rq import Worker, Queue, Connection

load_dotenv()
REDISTOGO_URL = os.getenv('REDISTOGO_URL')

listen = ['default']
redis_url = REDISTOGO_URL
conn = redis.from_url(redis_url)

if __name__ == '__main__':
    with Connection(conn):
        worker = Worker(list(map(Queue, listen)))
        worker.work()
```

Example Code for Enqueuing Tasks

```
task = q.enqueue(location_verification, scan_name, settings[0]['GEO_LOCATION'], job_id=title)
```

APPENDIX L – ETHICAL APPROVAL

Project Ethical Approval



Name: LIAM SHILLINGLAW

Project Title: Evaluating the Effectiveness of Various Developments to the Contactless 'Active' Reconnaissance Process

Reference: EMS3639

Status: Full Approval

Approval Date: 09.12.20

The Standard Conditions below apply to all approved student Research Ethics applications:

i. If any substantive changes to the proposed project are made, a new ethical approval application must be submitted to the Committee.
ii. The Proposer must remain in regular contact with the project supervisor.
iii. The Supervisor must see a copy of all materials and procedures prior to commencing data collection.
iv. Any changes to the agreed procedures must be negotiated with the project supervisor.

APPENDIX M – SETTINGS HANDLER

Code for Settings Handler

```python
# !/usr/bin/env python
# Name:     standardValues.py
# By:       LA-Shill
# Date:     21.04.2021
# Version   0.2
# ----------------------------------------------

import pymongo
from os import environ, path, getcwd
from dotenv import load_dotenv
from datetime import timedelta

"""Load in .env file"""
basedir = path.abspath(path.dirname(__file__))
load_dotenv(path.join(basedir, '.env'))

host = str(environ.get('DB_HOST'))
CORE_MONGO_DB = str(environ.get('CORE_MONGO_DB'))
VUL_MONGO_DB = str(environ.get('VUL_MONGO_DB'))
port = int(environ.get('DB_PORT'))

client = pymongo.MongoClient(host, port)
mainDB = client[CORE_MONGO_DB.rsplit('/', 1)[-1]]

try:
    if mainDB['settings'].find({}).count() > 0:
        print("[INFORMANT] Settings DB file detected")
    else:
        settings = {'CENSYS_API_ID' : '', 'CENSYS_API_SECRET' : '', 'SHODAN_API_KEY' : '', 'BINARY_EDGE_API_KEY' : '', 'ONYPHE_API_KEY' : '', 'FARSIGHT_API_KEY' : '', 'HIGH_RISK_PORTS' : [], 'GEO_LOCATION' : ''}
        mainDB['settings'].insert(settings)        Find related code in informant
except Exception as e:
    print("Error on first run, check you have all DB dependencies installed. Error: " + str(e))


class StandardValues:
    """
    Default vaules used across program,
    can be stored in volatile memory if required
    """
    db_r = mainDB['settings'].find({})

    CENSYS_API_ID: str = db_r[0]['CENSYS_API_ID']
    CENSYS_API_SECRET: str =  db_r[0]['CENSYS_API_SECRET']
    SHODAN_API_KEY: str =  db_r[0]['SHODAN_API_KEY']
    BINARY_EDGE_API_KEY: str = db_r[0]['BINARY_EDGE_API_KEY']
    ONYPHE_API_KEY: str =  db_r[0]['ONYPHE_API_KEY']
    FARSIGHT_API_KEY: str = db_r[0]['FARSIGHT_API_KEY']
    HIGH_RISK_PORTS: list = db_r[0]['HIGH_RISK_PORTS']
    GEO_LOCATION: str = db_r[0]['GEO_LOCATION']

    CENSYS_DEFAULT_RESULTS_QUANTITY: int = 100000
    CENSYS_FREE_PLAN_RESULTS_QUANTITY: int = 1000
    SHODAN_DEFAULT_RESULTS_QUANTITY: int = 10
    BINARYEDGE_DEFAULT_RESULTS_PAGE: int = 1

    THREATMINER_API_DELAY: int = 6
    ROBTEX_API_DELAY: int = 20
    THREATCROWD_API_DELAY: int = 10
    DALOO_API_DELAY: int = 5
    FARSIGHT_API_DELAY: int = 1

    DB_HOST: str = host
    DB_PORT: int = port
    MAIN_DB_NAME: str = CORE_MONGO_DB.rsplit('/', 1)[-1]
    VUL_DB_NAME: str = VUL_MONGO_DB.rsplit('/', 1)[-1]
```