An Empirical Exploration of Web Applications' Vulnerability Assessment: A Case Study Approach.

Danquah Paul Asante (Ph.D)
Council for Scientific and Industrial Research-Institute for Scientific and Technological Information (CSIR-INSTI)
Accra, Ghana
pauldanquah@yahoo.com

ABSTRACT

The security of web applications is predominantly under threat due to a myriad of security related vulnerabilities and threats to which they are subject. The assessment of these vulnerabilities are done either via automated methods or manual methods. This research work comparatively explores the use of various tools and methods for assessing the vulnerabilities of a web application. The case study used in this research is the IBM banking application specifically designed for demonstrating web application vulnerabilities (demo.testfire.net). This experiment used various simulated attacks to test whether the site has adequate security and the results are compared for the different tools and methods used. The benchmark used for assessment and comparison is the Online Web Application and Security Project. The findings tend to imply that proprietary tools are relatively more efficient than open source tools, however, due to the susceptibility of possible reports of false positives and false negatives, it is critically essential to confirm findings with manual testing methods. It is farfetched to assess all tools in one research exercise hence this research attempts to add to knowledge of comparing known web application vulnerability assessment tools in relation to contemporary vulnerabilities and methods for breaching web applications.

Keywords: Web Security, Vulnerability Assessment, Web Application Scanners

iSTEAMS Proceedings Reference Format
Danquah, P.A. (2019): An Empirical Exploration of Web Applications' Vulnerability Assessment: A Case Study Approach.
Proceedings of the 15th iSTEAMS Research Nexus Conference, Chrisland University, Abeokuta, Nigeria, 16th – 18th April, 2019. Pp 1-10.
www.isteams.net DOI Affix - https://doi.org/ 10.22624/AIMS/iSTEAMS-2019/V15N1P1

1. INTRODUCTION

The use of web applications have grown over the years, so has the security related incidents of web applications. Web applications are predominantly being used as communication channels between service providers and clients. Web applications predominantly contain client side scripting code such as HTML, JavaScript and Cascading Style Sheets. The web applications would typically also contain server side scripts such as PHP, JSP, C with embedded SQL queries that are integrated to database management systems such as Oracle, Ms SQL, My SQL etc. Contemporary web applications include technologies such AJAX (Asynchronous java script and XML), this enhances web application and gives more interaction and response to the user. Numerous vulnerabilities in web application security such as cross-site scripting, SQL injection and cross-site request forgeries are well known problems reported with thousands of web applications annually. The extension of corporate portals for various transactions have been largely successful but not without known security risks. The desire to stay ahead of the competition while minimizing cost by leveraging technology means the process is driven by pressure to achieve results.

There has also been a growth in the automation of vulnerability assessments and penetration test of web applications. "The evolution of the Internet has brought a significant change in the evolution of software, resulting in an increasing presence of Information Systems in Web environments and, consequently, an increase in security vulnerabilities and threats" (Leite & Albuquerque, 2019). Various researchers have attempted assessing a myriad of specific vulnerabilities as well as the respective tools used to assess these vulnerabilities. Bairwa, Mewara and Gajrani (2014) showed that "different scanners detect different types of vulnerabilities but a single tool is not capable of detecting all type of vulnerabilities". A broad consensus about the most critical security risks to web applications and a standardized approach to an unbiased source of information on best practices is the Online Web Applications Security Project (OWASP). Much as it is impossible to assess all tools in one research exercise, the attempt in this research is to add to knowledge by comparing some known assessment tools in relation to contemporary vulnerabilities and methods for breaching web applications.

2. LITERATURE REVIEW

Vulnerability Assessment is the art of finding an open door. Penetration Testing involves a series of activities undertaken to identify and exploit security vulnerabilities. Penetration testing is widely used to help ensure the security of the network. "Traditional penetration testing were manually performed by tester according to scheme, the process is usually complex resulting in that it is labor-intensive and requires tester to be familiar with all kind of tools".(Umrao,Kaur & Gupta, 2018). Muñoz, Cortes & Villalba (2018) instructively explained that "there are two main kinds of vulnerable web applications, usual applications developed with a specific aim and applications which are vulnerable by design. On one hand, the usual applications are those that are used everywhere and on a daily basis, and where vulnerabilities are detected, and often mended, such as online banking systems, newspaper sites, or any other Web site".

On the other hand, vulnerable by design web applications are developed for proper evaluation of web vulnerability scanners and for training in detecting web vulnerabilities. The main drawback of vulnerable by design web applications is that they used to include just a short set of well-known types of vulnerabilities, usually from famous classifications like the OWASP Top Ten. Martirosyan (2012). The concept of security in web applications is not new. However, it is often ignored in the development stages of the applications. Being multitiered and spread across different domains, it is challenging to come up with a security solution that works for all web applications. Moreover, developers are more inclined to implement features and often do not practice secure coding. Anis, Zulkernine, Iqbal, Liem & Chambers (2018).

"The main motive of the vulnerability scanner is to identify the vulnerability and produce a better result/report of each web application in an effective manner". Durai & Priyadharsini (2014). While developing the websites, many times developers/site owners forget to remove sensitive data from website which is not supposed to be exposed to public users. Such data consists of untested vulnerable forms, database backup, and site backup in compressed format. This gives a hacker the opportunity to search for that kind of data and use it adversely (Patil & Gosavi, 2015). Open source web application vulnerability scanners include OWASP ZAP, Burpe Suite, Nikto and Nessus, other proprietary web application scanners are AppScan, Netsparker, Acunetix and RETINA. Most of the tools have the inbuilt capability to scan for relatively current vulnerabilities in web applications, these vulnerabilities are then presented in a report format based on the standard required such as Online Web Application Security Project (OWASP), Payment Card Industry – Data Security Standard (PCI-DSS) and Web Application Security Consortium (WASC). The most generic and widely adopted standard in scanning and reporting is the OWASP top 10 approach which is periodically released.

The OWASP Top 10 - 2017 is based primarily on data submissions from firms that specialize in application security and an industry survey that was completed by over 500 individuals. This data spans vulnerabilities gathered from hundreds of organizations and over 100,000 real-world applications and APIs. The Top 10 items are selected and prioritized according to this prevalence data, in combination with consensus estimates of exploitability, detectability, and impact. (OWASP Top 10, 2017)

Zhang & Zhang, (2018) posited that at present, SQL injection attacks have become the main method of hacking. SQL injection vulnerabilities seriously threaten the security of WEB application systems. This was confirmed via a security assessment of web applications through penetration system techniques by Lubis & Tarigan (2017).

Phanindra, Narasimha & PhaniKrishna, (2019) proposing a non-functional requirement Application Security Management is to define the requirements for security in all applications that use the web application security standards. Web application security has turned into the essential talk for security specialists, as application assaults are always on rise and posturing new dangers for associations. An approach for the treatment of vulnerabilities in Web applications composed by the selection and mapping of categories of major vulnerabilities, risks and existing proactive controls to prevent or treat occurrences of exploitation of vulnerabilities by malicious agents is also proposed by Ayachi, Ettifouri, Berrich & Toumi (2019)

In an experiment by Sarfaraz & Khan (2018) including 28 search phrases, achieved 67.86% hit rate at first guess and 96.43% hit rate within three guesses. The results showed that HTTPS traffic can be correlated and de-anonymized through HTTP traffic and secured search are not always secure unless HTTPS by default enabled everywhere. In the research that assessed vulnerability scanners' proactive approach to assess web application security, a comparison of various tools produced the details in figure 1 below (Bairwa, Mewara and Gajrani 2014);

Table 1: Vulnerabilities

Vulnerabilities	Nmap	Nessus	Acunetix WVS	Nikto	BurpSuite
SQL Injection	√	√	√		√
Improper Error Management	√	√	√		√
Cross site Scripting	√	√	√	√	√
Rogue Servers	√	√		√	
Denial of Service	√	√	√		√
Remote Code Execution		√			
Format String Identifier		√	√		√
IIS.printer		√	√		√
DCOM					

Source: (airwa, Mewara & Gajrani (2014)

The challenge with this output is the tools used are not all specialized web application vulnerability scanners. They are predominantly generic vulnerability assessment applications or tools.

OWASP Top 10

The OWASP Top 10 is based primarily on information from firms that specialize in application security and an industry survey, the aim of the OWASP Top 10 is to educate developers, designers, architects, managers, and organizations about the consequences of the most common and most important web application security weaknesses. It also provides basic techniques to protect against these high risk problem areas, and provides guidance on how to resolve the security lapses. : Below is a listing of the OWASP top 10 for 2013 and 2017;

Table 2: Listing of the OWASP top 10 for 2013 and 2017;

OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	↘	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	∪	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↘	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	∪	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	☒	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	☒	A10:2017-Insufficient Logging&Monitoring [NEW,Comm.]

Injection: This occurs when malicious data is sent to interpreters as part of query to gain unauthorized access to data. Broken Authentication: This is where confidential data such as passwords, keys or session tokens are compromised for the purpose of assuming other users' identity. Sensitive Data Exposure: This occurs where sensitive data is compromised without appropriate encryption of data either at rest or in transit. XML External Entities (XXE): This vulnerability is experienced where poorly configured documents leverage on external entities to obtain information on internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

Broken Access Control: Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc. Security misconfiguration: This vulnerability is experienced as result of typically insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Cross-Site Scripting (XSS): This vulnerability is experienced when an application includes untrusted data in a new web page without proper validation or escaping, or updates an existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

Insecure Deserialization: This vulnerability occurs when there are deserialization flaws that can be exploited to perform attacks including replay attacks, injection attacks, and privilege escalation attacks. Using Components with Known Vulnerabilities: This is experienced when components such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts. Insufficient logging and monitoring coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

Hall and Watson (2016) described various vulnerability assessment and penetration types and methods, these, instructively include Black Box, Gray Box, and White Box. Furthermore, Internal and External Penetration tests are also deliberated upon. The white box approach is described as a method in which the penetration tester is given the complete knowledge of the target. The black box on the other hand is when the attacker has no knowledge of the target, when the tester is given partial information about the target, this is referred to as gray box penetration testing. In the context of penetration test, if the test is conducted from outside the network, this is referred to as external penetration testing whereas If the attacker is present inside the network, this is referred to as internal penetration testing.

3. METHODOLOGY:

This research is qualitative research in nature with an attempt to gather non-numerical data. A case study involving an experimental validation is used. The research uses a publicly accessible web application http://demo.testfire.net as the target for assessment with inherent vulnerabilities. The website is published by IBM Corporation for the sole purpose of demonstrating the effectiveness of products in detecting web application vulnerabilities and website defects. Much as it looks like a banking site, it is not a real banking site. It is setup without warranties of any kind, either express or implied.

A total of four tools are used to assess the vulnerabilities on this site via automated scans, the tools consist of two proprietary scanners and two open source scanners. The proprietary ones are namely acunetix and netsparker. The open source scanners are namely OWASP ZAP and Vega. Manual methods were also used to confirm the critical and high risk vulnerabilities identified. The focus is to scan for relatively contemporary vulnerabilities that are exploited with emphasis on predominantly OWASP top 10 for the years 2013 and 2017. The reporting of findings focuses on using the Common Vulnerability Scoring System (CVSS) and Common Vulnerabilities and Exposures (CVE) system. The CVSS is a free and open industry standard for assessing the severity of computer system security vulnerabilities.

CVSS attempts to assign severity scores to vulnerabilities, allowing responders to prioritize responses and resources according to threat. Scores are calculated based on a formula that depends on several metrics that approximate ease of exploit and the impact of exploit. Scores range from 0 to 10, with 10 being the most severe. While many utilize only the CVSS Base score for determining severity, temporal and environmental scores also exist, to factor in availability of mitigations and how widespread vulnerable systems are within an organization, respectively. The Common Vulnerabilities and Exposures (CVE) system on the other hand provides a reference-method for publicly known information-security vulnerabilities and exposures. The CVSS v3.0 Qualitative Severity Rating Scale is shown below;

Table 3: Source: CVSS v3.0 Specification Document

Rating	CVSS Score
None	0.0
Low	0.1 - 3.9
Medium	4.0 - 6.9
High	7.0 - 8.9
Critical	9.0 - 10.0

Blockmon (2016) outlines the stages involved in vulnerability assessment and penetration testing as namely;

- Reconnaissance
- Scanning
- Gaining Access
- Maintaining Access
- Covering Tracks

The phases are further confirmed by Sabah (2018) and explained, the reconnaissance phase is when an attacker attempts to gather information about a target before launching any attacks. This information being gathered could span employees, networks, systems as well as any relevant third-parties. The reconnaissance is categorized into the passive and active reconnaissance. Passive reconnaissance involve obtaining information without direct interaction with the target whereas active reconnaissance involves direct interaction with the target.

The scanning phase involves the use of tools to determine live systems for the purpose of enumerating the vulnerabilities and planning how to evade any security measure in place. The gaining access phase is when the attacker successfully gains access to systems as a result of vulnerabilities observed. Various methods may be used to gain access and subsequently exploit the access obtained. The fourth phase which involves maintaining access refers to retaining control of the target system. Typically this is done with methods and tools that enable the attacker maintain access in stealth mode. The last phase in is considered unethical for authorized vulnerability assessment and penetration testers. This is the phase where the attacker covers their tracks by using various methods to avoid detection.As an ethical hacker, it is important that you know these same steps that an attacker uses in order to play a huge role in defending against cyber-attacks.

4. RESULTS

Acunetix Results Summary:

Figure 1: Results from Acunetix Scan

The listed critical and high risk identifications via acunetix are namely;

- Cross site scripting (verified)
- Directory traversal
- DOM-based cross site scripting
- Microsoft IIS tilde directory enumeration
- SQL injection
- Application error message
- Backup files
- Basic authentication over HTTP
- Directory listing
- HTML form without CSRF protection
- User credentials are sent in clear text

Netsparker Summary Results

Table 4: Results from Netsparker Scan

	ISSUES	INSTANCES	CONFIRMED
CRITICAL	1	2	0
HIGH	4	5	5
MEDIUM	4	5	5
LOW	10	10	3
INFORMATION	6	6	3
TOTAL	**25**	**28**	**16**

- Cross-site Scripting *Password Transmitted over HTTP
- Cookie Not Marked as Secure *Local File Inclusion
- SQL Injection

OWASP ZAP Summary Results;

Table 5: Results from OWASP Zap scan

Risk Level	Number of Alerts
High	3
Medium	4
Low	152
Informational	61

- Cross-site Scripting
- Directory listing
- User credentials are sent in clear text

VEGA Results Summary

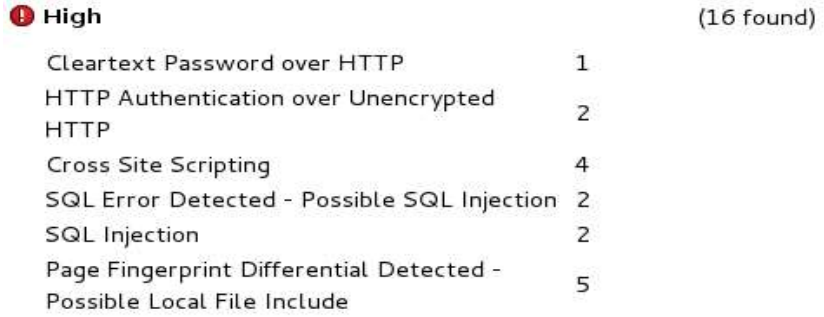

Figure 2: Results of Vega Scans

Manual Results;

- Injection
- Cross-Site Scripting (XSS)
- Cross Site Request Forgery
- Directory Listing
- Broken Authentication and Session Management

Table 6: Comparative Results of Various Scans and Tests

No	Vulnerability	Acunetix	Netsparker	OWASP ZAP	Vega	Manual
1	Injection	X	X	X	X	X
2	Cross Site Scripting	X	X	X	-	X
3	Clear Text Transmission	X	X	X	X	X
4	Directory Listing/Traversal	X	-	X		X
5	Cross Site Request Forgery	X	-	-	-	X

The comparative analysis of the methods of assessment show that the usage acunetix tool and manual methods are technically the most reliable in detecting vulnerabilities of web applications. OWASP Zap is also relatively more reliable as compared to Netsparker and Vega though less reliable than acunetix and manual methods. It however needs to be emphasized that, this is based on the assumption that indeed the OWASP top 10 vulnerabilities are the vulnerabilities of most essence.

5. DISCUSSION

The findings from this research show that the various tools used are not necessarily adhering strictly to the Common Vulnerabilities and Exposures (CVE) standard in assessing vulnerabilities in web applications. There also tends to be significant difference in the total number of identified vulnerabilities by each method. Another key observation is the disparity in the identified vulnerabilities in relation to the Online Web Application Security Project's top 10 vulnerabilities. Not all tools identified the vulnerabilities within the top 10 known vulnerabilities, it could however not be wrong to imply form findings as shown in comparison table that acunetix as a tool provides a relatively more reliable feedback on vulnerabilities. The manual methods of assessing vulnerabilities also proved to be quite reliable given the relative confirmatory feedback obtained as compared with the other tools.

The findings depict a perception that supposed proprietary proprietary tools such as acunetix and netsparker tend to be more reliable and exhaustive in its probe for vulnerabilities within web applications. A significant number of the findings of these tools were accurately confirmed with manual methods of querying the application with manual commands and methods. While these are findings and possible deduction from the findings, it is essential to note that various subsequent versions of the tools may possibly make up for the limitations of the used versions. Martirosyan(2012) carried out an extensive analysis of WAVS running results and was presented for each OWASP 2013 Top Ten vulnerability type. Their research traced each attack to the root in order to interpret and judge the result and, more importantly, find the cause of False Negative vulnerabilities. Comparatively, it was discovered in their research that several vulnerabilities, for instance CSRF, were missed because of incomplete crawling functionality of the scanners. Patil and Gosavi(2015) also evaluated different types of vulnerabilities and found that different methods are required for to detect. In addition, various web scanners are trying to automate the detection as much as possible, however, it is always best to confirm vulnerabilities by manual work to ensure false positives are reduced or completely eliminated.

In relation to the findings of this research, it is worth noting that Martirosyan (2012) also indicated that "acunetix WVS showed brilliant results for discovering the AJAX flaws, the detection rate was 100%. It also had a very good detection rate of other types of XSS vulnerability". The intriguing finding in their research was the fact that, at the same time, though, the acunetix WVS false positive rate for cross site scripting vulnerabilities was reported as 205.5%, this is relatively high. It must be emphasized that this research did not delve into addressing false positives of findings from scanners, it only attempted to confirm OWASP related findings via manual methods. The results are shown in table 1. The clinical relevance of the findings of this research imply there proprietary web application vulnerability scanners tend to identify more industry standard results of web application vulnerabilities. Given the fact that there are significantly many more of such scanners, there is the need to indicate that this research work is definitely not exhaustive. In addition, the platform for the case study was developed with Java, it would be prudent for similar research work to be carried out on platforms developed with other languages such as PHP, ASP etc.

6. CONCLUSION

Web applications are exposed to a myriad of security related vulnerabilities and threats. This paper explored various vulnerability scanners for web applications as well as manual methods. Reports from different scanners produce different test results. It is expected that a good scanner should find as many numbers of vulnerabilities as possible. The assessment of these vulnerabilities are done either via automated methods or manual methods. The subject of the study which was IBM designed banking application specifically designed for demonstrating web application vulnerabilities (demo.testfire.net) had inherent weaknesses. This research concludes that the use of scanners is really good, however, the type of scanner used determines the nature of results obtained. It is essential to always verify identified vulnerabilities from scanners manually to avoid reporting false positives. Further research is needed for identifying which scanners are most appropriate for the respective platforms on which web applications run. It is quite farfetched to assess all tools in one research exercise hence this research has objectively contributed to knowledge of comparing known web application vulnerability assessment tools in relation to contemporary vulnerabilities and methods for breaching web applications.

Future Work

Pursuant to the findings from this empirical research, it is recommended that a relatively thorough assessment is done on comparing open source and proprietary tools for efficiency, effectiveness and reliability. The platform for development of the test web application would also need to be assessed to determine if there is any specific capabilities or limitations with open or closed source tools on specific web application platforms.

REFERENCES

1. Ayachi Y., Ettifouri E.H., Berrich J. & Toumi B. (2019). Modeling the OWASP most critical web attacks. Smart Innovation, *Systems and Technologies* 111, pp. 442-450
2. Bairwa S., Mewara B. & Gajrani J. (2014), Vulnerability scanners: A Proactive Approach to Assess Web Application Security*, International Journal on Computational Sciences & Applications (IJCSA)* Vol.4, No.1, February 2014
3. Blockmon R. (2019). CEH V9: Certified Ethical Hacker Version 9 Practice Tests, *Wiley*, ISBN 978-1-119-25224-5
4. Durai K.N & Priyadharsini K. (2014). A Survey on Security Properties and Web Application Scanner, *International Journal of Computer Science and Mobile Computing*, ISSN 2320–088X, Vol. 3, Issue. 10, October 2014, pg.517 – 527
5. Hall G. & Watson E. (2016). Hacking: Computer Hacking, Security Testing, Penetration Testing, and Basic Security, *Create Space Independent Publishing Platform*, ISBN 1541289323
6. Harshala P. Patil P. & Gosavi B. (2015). Web Vulnerability Scanner by Using HTTP Method*, International Journal of Computer Science and Mobile Computing*, Vol.4 Issue.9, pg. 255-260
7. Leite G.S. & Albuquerque, A.B. (2019). An approach for reduce vulnerabilities in web information systems, *Advances in Intelligent Systems and Computing*, 860, pp. 86-99
8. Lubis A. & Tarigan A. (2017). Security Assessment of Web Application Through Penetration System Techniques, *International Journal of Recent Trends in Engineering & Research (IJRTER)* Volume 03, Issue 01; January - 2017 [ISSN: 2455-1457]
9. Martirosyan Y. (2012). Vulnerability Scanners' Strengths and Limitations Using Custom Web Application, *California State University Press*, East Bay
10. Nasir A., Arshah R.A., Hamid M.R.A. & Fahmy S. (2019). An analysis on the dimensions of information security culture concept: A review, *Journal of Information Security and Applications* 44, pp. 12- 22
11. Online Web Application Security Project Top 10, 2017.
12. Phanindra R., Narasimha A. & PhaniKrishna V.B (2019). A review on application security management using web application security standards, *Advances in Intelligent Systems and Computing*, 731, pp. 477-486
13. Sabih Z. (2018). Learn Ethical Hacking from Scratch: Your Stepping Stone to Penetration Testing, *Packt Publishing*, ISBN 139781788622059
14. Umrao S., Kaur M. & Gupta G. K. (2012). Vulnerability Assessment and Penetration Testing, *International Journal of Computer & Communication Technology* ISSN (PRINT): 0975 - 7449, Volume-3, Issue-6, 7, 8, 2012
15. Yin S., Sheng, J., Wang, T. & Xu H. (2019). Analysis on mobile payment security and its defense strategy *Advances in Intelligent Systems and Computing*, 773, pp. 941-946

About the Author

Dr. Paul Asante Danquah is an IT professional, Lecturer and a Research Scientist who holds a BSc HONS in Computing, MSc in Information Security and a PhD in IT (Specialized in Cybercrime) from the University of Greenwich UK, Anglia Ruskin University UK and Open University Malaysia respectively. He has various industry certifications, some of which are ISO 27001 Lead Implementer, Certified Ethical Hacker (CEH), Data Center Infrastructure Expert (DCIE), Cisco Certified Network Professional (CCNP) and Microsoft Certified Systems Engineer (MCSE). Dr. Paul Danquah has worked in various capacities over the last 20 years, these range from Programmer, Network Engineer, IT Manager and Technical Director of various organizations.