ISSN 2278-3091
Volume 8, No.1.2, 2019
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse1981.22019.pdf
https://doi.org/10.30534/ijatcse/2019/1981.22019

BIO-INSPIRED DEPENDABILITY ANALYSIS OF SECURITY IN SOFTWARE DEVELOPMENT LIFE CYCLE PROCESS

Saleem Basha[1], Gazala yusufi[2], Rajbunisa[3], Abbas[1], Saravana Balaji B[4]
[1] Assistant Professor, Mazoon College, Oman, m.s.saleembasha@gmail.com
[2] Lecturer, Mazoon College, Oman
[3] M.Tech Student, Mazoon College, Oman, rajbunisa@gmail.com
[4] Assistant Professor, Department of Information Technology, Lebanese French University, Erbil, KR-Iraq, saravanabalaji.b@gmail.com

ABSTRACT

Security analysis of software is analyzed by a close exploration of the modules that were developed during the Software Development Life Cycle (SDLC) process. These analyses are grabbing the importance of designing the complex and composite software systems. Standardized methodologies and tools were available for designing highly complex software system. But very fever tools are available for calibrating the dependability analysis of security. The work in this paper is to stanch the definition of security-related dependability modeling. This modeling could be used to capture the dependability attributes like security, reliability, and availability in the preliminary phases of the complex software system. This insight provides the guidelines to choose the appropriate architecture and design solutions.

Key words: Bio Inspiration, Dependability Analysis, Security, SDLC.
\

1. INTRODUCTION

Evolution of dependability analysis of security is a critical consideration to assess whether the enterprise information system is being developed satisfies its target. Analytical modeling has proven to be useful and versatile to evaluate these attributes in the design phase. Dependability models allow comparing different architectural solutions and design choices and to run sensitive analysis identifying both dependability bottlenecks and critical parameters to which the system is sensitive [1]. The ability and timely services of the software system can be calibrated form its dependability and its security. Dependability is the ability to deliver service that can be described in terms of its dependability and security and it is the ability to deliver service that can justifiably be trusted, and can be stated as an integrative concept that encompasses the attributes availability, reliability, safety, integrity and maintainability [2]. Security, on the other hand, is defined as a concept addressing the attributes confidentiality, integrity and availability [3]. Regardless of the circumstances that an enterprise information system cannot be advocated that it is trustworthy without a thorough analysis of dependability of security. The traditional dependability analysis techniques have made the analyst to lean towards probabilistic modeling, which is used to offer quantitative calibration of the operational security over the enterprise information system. However, most of the research has focused on security or dependability analysis. This paper extends our previously published work by integrating the proposed model in a web service computing environment. The information system is the avenue to use and exchange enormous amount of confidential data such as passwords, credit card number, insurance number etc. across the networks and are porn to vulnerable due to the complexity of the system and time to market pressure the system developers are unaware of the security breaches, the most of the systems (90%) [4]. suffer from errors that make the possible breaking of confidentiality, integrity or availability of delivered services. These kinds of vulnerability can be eroded by a thorough analysis of the security dependencies**.**

2. BACKGROUND WORK

There are many related works in the context of this study. In the current scenario, many new developments like big data, cloud computing are having security issues which are discussed in [12-15]. Different authors have explored numerous methods of incorporating security in the system development life cycle process. A brief outline of some of the related background work is as follows: Shanmuga Priya S., et al. [5] have explored the possibility of having security during the entire SDLC process such that security attacks can be resisted and major flaws in the system development process can be prevented. The authors have discussed the different rules to be followed by all stakeholders during the SDLC process such that vulnerabilities can be avoided. They have tried to apply security mechanisms in the requirement elicitation, system design, implementation as well as the testing phases by using the threat modeling concept which enabled them to find the threats in each phase and later map it to the security policies. Cho C., et al. [6] have investigated the

prospects of subsuming cyber-physical security and dependability analysis in digital control systems of nuclear power plants. Cyber-physical security, as well as dependability, are critical issues which need to be considered for the safety of NPP's. Therefore, the authors came up with a framework which could prevent cyber-attacks as well as conform to the cyber security regulations. Simultaneously they have also proposed a physical framework for physical attacks. Then they also go through the dependability analysis to suggest that the cyber framework designed is highly dependable. They have also discussed in detail about the security of control systems in NPP's covering all the past accidents taken place due to security slacks, and how the evolving security regulations have helped to combat the problems related to physical security only whereas the cyber security was not much evolved. A case study helped them to explain the cyber-physical security slacks better and develop cyber security framework conforming to the cyber security standard RG 5.71 and preventing the possibility of outside intrusion. Jarzombek J., et al. [7] in their paper have discussed the security in the software life cycle and dependability factors like quality and reliability during development and deployment. This paper includes various tools and practices that software developers need to consider in order to diminish the possibilities of security attacks and failures during the SDLC process. The authors talk about shifting the focus of development towards security which changes the life cycle in a better way. Risk-driven requirement engineering that uses threat modeling can be beneficial for the developers. Security-enhanced process models were demonstrated to improve the efficiency and adaptability of the SDLC activities and minimizing the number of errors. Thus, this work provides developers with a two-phase security enhancement process. Phase 1- showing them the important security practices to be used throughout the development and Phase 2- gives an idea about how to provide an increased level of security keeping room for improvement. Thiriet J., et al. [8] have delved in detail on the dependability issues cyber security of cyber-physical systems. This paper presents the different problems that can affect the cyber security and how to eradicate them. Further, they consider the various dependability factors which need to be standardized so that the systems under consideration may work as expected. Then they move towards risk analysis and discuss the factors that need to be taken into account for cyber security. Intrusion detection systems (IDS) usage for protection of IT infrastructure is also dealt with coming to a conclusion that processes need to be controlled and potential vulnerabilities need to be taken into account while dealing with cyber-physical systems. Assal H., et al. [9] in their paper have addressed the security practices needed to be considered in the SDLC processes but their approach was different from other authors in a sense that they conducted a series of interviews with developers to investigate the security practices used by them during the SDLC phases. Then they compared these real-life security practices with the works of many authors and thus came to the conclusion that they differed from the literature survey conducted. This difference was attributed to the complex and heavy-weight procedures used as best practices which discouraged developers from using them; thus leading to security problems. So, the study suggested the need for new, light-weight best practices which release the burden of security maintenance in the system development process. Chang X., et al. [10] has developed a survivability model for security and dependability analysis of a vulnerable critical system. Their paper suggests a model and metrics which can not only capture the vulnerable system behavior but also find the survival attribute of the system in terms of security risk and dependability. The model, metrics and numerical results presented in this paper suggest the various investment efforts that can be used on the system recovery strategies.

3. DEPENDABILITY ANALYSIS

Security analysis of software is analyzed by a close exploration of the modules that were developed during the Software Development Life Cycle (SDLC) process. This research article mainly maps the software dataset data to the equations by Yukihiro Chiba and Kichiro Shinozaki [11]. Mapping is done as follows in table 1.

Table 1: Mapping

Stem Density (S)	Security (δ)
Position (z)	Module (m)
Change (Δ)	Change (Δ)
Cumulative Stem Increment (CSI)	Shirked Security Aspect (SSA)

The analysis begins with assuming the mapping variable in the above table to yield the high convergent rate. For understanding a short description is given with respect to the above mapping variable to the main equations in [11]. The initial assumption is taken by slicing the security aspect of software module into many pieces of constant lines of code (LOC), with respect to security consideration the weight of each piece is denoted as the security (δ). In addition to that, shirked security aspect could be accommodated in the later stage of the module may be defined as the Shirked Security Aspect (SSA). The variables are defined as follows: 't_0' is the time when the software development was completed, 'm' is the module different between the initial modules and the final modules at time t_0, δ(m) is the security of the module *m* at time t_0 and Δδ(m) is the inclusion of shirked security aspect in the later stage of software development of δ(m). The quantitative analysis would be difficult because the relationships among δ(m) and Δδ(m) diverge so extensively due to the values of $\Delta\delta(m)$ in particular module are uncertain due to the functionalities of that modules. In general, the shirked security aspect with respect to the particular

module can be formulated and defined as "Shirked Security Aspect (SSA)" as

$$SSA(\delta) = \int_0^m \Delta\delta(m)\,dm \quad (1)$$

From the eqn(1), the relationship can be assumed as

$$\int_0^m \Delta\delta(m)\,dm = k\delta(m) \quad (2)$$

where k is a proportionality constant.

It is well known that, with respect to the arbitrary function $\emptyset(x)$ of a variable '*x*'. It was assumed that $\frac{\partial k}{\partial t} = 0$ for arriving the Eqn(3)

$$\delta = \delta(t, m) = \emptyset(kt + m) \quad (3)$$

The left hand side expression is the function of time 't' and the right hand side expression is the function of module 'm'. These two expressions are equal to the proportionality constant 'k'

$$\frac{1}{kT}\frac{dT}{dt} = \frac{1}{M}\frac{dM}{dm} = K\,(\text{constant}) \quad (4)$$

Where 'm' is the module difference between the module 'm' to the final module at time t_0, similarly, the security at the same module 'm' at time t_0- t_1, t_1 >0) is

$$\delta(t_0 - t_1, m) = \emptyset(kt_0 - kt_1 + m) \quad (5)$$

where, t_1 is the time taken to develop a set of modules after module 'm' was completed. Static Application Security Testing (SAST) approach is used to collect the data for an in-house software test case.

Table 2: The test cases are performed for the following parameters

Sl No	Parameters	Sl No	Parameters
1	Total number of test cases	20	Dependent defect
2	Number of test cases passed	21	Number of variables
3	Number of test cases failed	22	Time to confirm a bug
4	Number of test cases blocked	23	Access control issues
5	Number of defects found	24	Breaches
6	BOON	25	Out bond
7	CQual	26	In bond
8	Perl's taint	27	Defect injection rate
9	Security Value	28	Defect distribution by module
10	Cognition	29	Escape sequence
11	Risk	30	vulnerability
12	Responsibility	31	Exposures
13	Secure Coding	32	Variables
14	Compliance	33	Memory access
15	Peer influence	34	Authentication
16	Expectation	35	Authorization
17	Exposed	36	Availability
18	Fixed defects	37	Non repudiation
19	Static defects	38	Confidentiality

Table 3: Sample Dataset for Test Run

Total number of test cases	Number of test cases passed	Number of test cases failed	Number of test cases blocked	Number of defects found	BOON	CQual	Perl's taint	Risk	compliance	Expectation	Static defects	Dependent defect	Out bond	In bond	Peer influence	Exposed	Fixed defects	Number of variable	Breaches	[illegible] distribution by [illegible] vulnerability	Exposures	Variables	Memory access	Escape sequence	Total	Module (m)	Per Module	Security (δ)	Change (Δδ)	SSA	Gross SSA
386	277	45	32	32	106.29	0.32	3.61	0.97	0.35	0.88	0.08	0.47	0.78	0.72	23	27	625	315	149	14	871	877	27	585	94	1.22E-05	74	1.65E-07	0.42	3.92E-07	0.001	0.41
481	363	32	48	38	108.63	0.51	5.65	0.71	0.64	0.33	0.53	0.96	0.38	0.94	58	76	116	405	240	366	275	6	331	673	68	1.23E-05	117	1.05E-07	0.90	1.17E-07	0.001	0.38
406	313	35	25	33	102.83	0.92	3.03	0.77	0.52	0.40	0.67	0.34	0.76	0.00	37	30	443	739	605	731	925	668	148	704	49	5.84E-05	14	4.17E-06	0.86	4.87E-06	0.000	0.19
459	344	42	23	50	101.03	0.45	4.55	0.30	0.45	0.16	0.91	0.76	0.82	0.67	75	38	882	458	520	653	467	814	475	602	33	0.000115	58	1.98E-06	0.75	2.64E-06	0.004	2.04
408	292	23	43	50	101.94	0.75	5.41	0.39	0.40	0.05	0.26	0.59	0.47	0.09	4	83	231	828	781	289	850	724	558	355	29	0.000164	97	1.69E-06	0.65	2.59E-06	0.012	4.96
396	300	27	28	41	105.02	0.24	5.22	0.14	0.41	0.26	0.14	0.76	0.61	0.15	66	26	813	847	799	102	484	520	364	785	71	2.69E-05	14	1.92E-06	0.02	0.000123	0.012	4.76
355	255	44	25	31	105.63	0.34	5.88	0.27	0.67	0.61	0.61	0.65	0.42	0.65	9	8	647	288	390	712	232	13	246	634	40	6.45E-05	137	4.71E-07	0.76	6.22E-07	0.006	2.07
375	285	27	39	24	108.85	0.56	5.07	0.57	0.51	0.07	0.08	0.85	0.59	0.36	12	68	634	237	364	604	709	454	972	405	72	2.5E-05	124	2.02E-07	0.74	2.71E-07	0.002	0.78
488	366	48	49	25	107.79	0.93	4.18	0.26	0.77	0.90	0.99	0.50	0.82	0.50	31	14	31	109	795	980	241	293	401	87	81	1.01E-05	68	1.49E-07	0.74	2.01E-07	0.000	0.23
396	273	49	31	43	104.21	0.88	4.56	0.38	0.53	0.91	0.80	0.27	0.09	0.02	96	3	739	710	199	210	620	587	830	594	45	7.17E-05	159	4.51E-07	0.56	8.02E-07	0.010	4.01
481	359	42	38	42	101.16	0.12	3.77	0.67	0.78	0.43	0.82	0.88	0.96	0.19	15	74	114	302	141	589	673	962	421	206	45	4.17E-05	171	2.44E-07	0.49	5.02E-07	0.007	3.53
433	312	47	50	24	100.79	0.44	3.09	0.33	0.91	0.18	0.27	0.08	1.00	0.81	46	72	759	732	485	470	617	775	967	109	45	7.06E-05	57	1.24E-06	0.39	3.15E-06	0.005	2.21
443	351	42	23	27	109.81	0.21	3.65	0.26	0.57	0.66	0.64	0.73	0.61	0.15	31	75	350	873	277	24	606	838	737	570	92	1.19E-05	44	2.7E-07	0.12	2.32E-06	0.002	1.00
405	302	38	27	38	109.60	0.31	3.11	0.36	0.53	0.47	0.94	0.38	0.61	0.73	93	16	964	204	41	340	334	541	756	564	55	3.44E-05	93	3.69E-07	0.09	4.04E-06	0.017	7.08
346	235	48	31	32	106.46	0.72	4.96	0.73	0.18	0.06	0.21	0.65	0.80	0.43	23	60	945	496	716	461	236	53	753	360	50	5.75E-05	181	3.18E-07	0.36	8.78E-07	0.014	4.98
464	366	31	26	41	107.73	0.80	3.84	0.61	0.09	0.11	0.45	0.99	0.15	0.51	80	84	0	740	375	570	174	161	757	560	28	0.000102	67	1.52E-06	0.90	1.69E-06	0.004	1.76
344	249	33	26	36	107.96	0.73	3.64	0.70	0.22	0.74	0.19	0.87	0.59	0.34	0	25	697	642	806	236	862	646	6	916	67	3.59E-05	51	7.05E-07	0.34	2.1E-06	0.003	0.94
371	268	35	21	47	101.18	0.62	4.06	0.62	0.65	0.11	0.16	0.73	0.18	0.13	95	47	469	228	743	504	660	828	137	567	91	1.69E-05	43	3.94E-07	0.79	4.99E-07	0.000	0.17

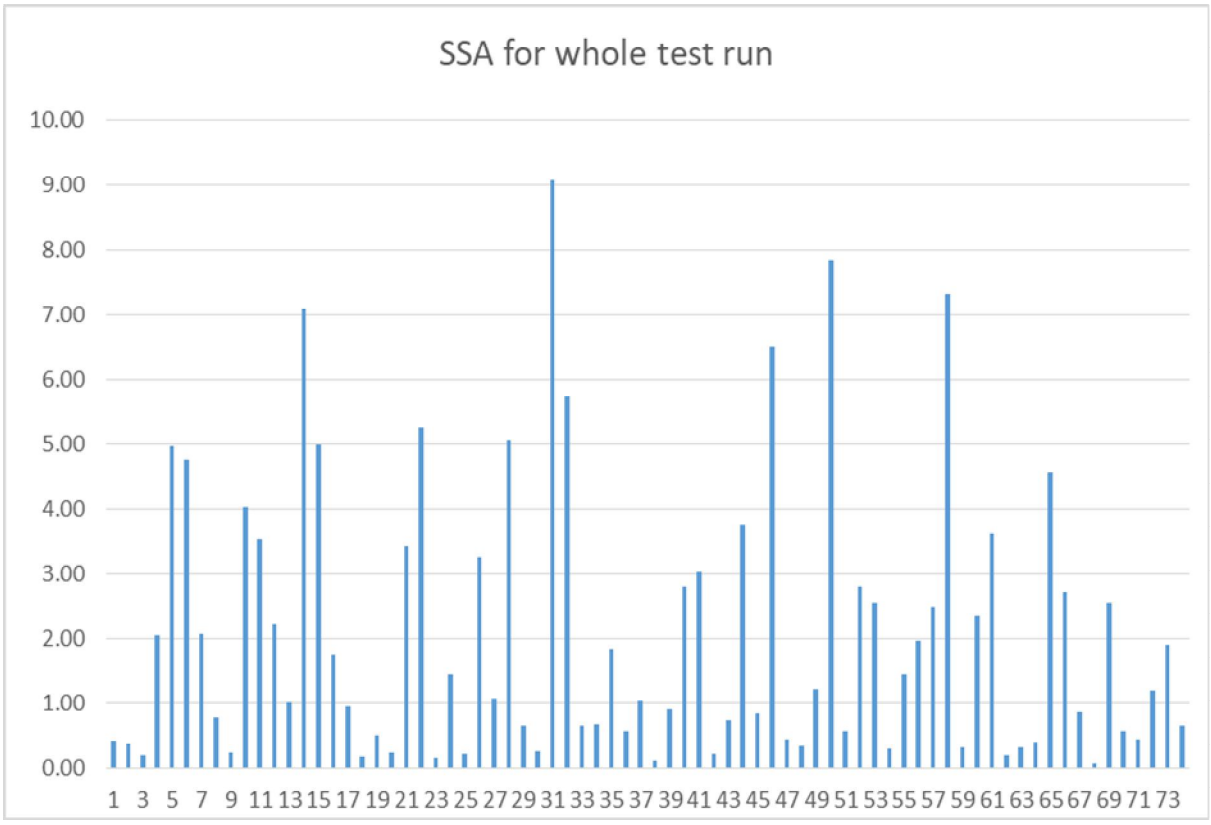

Figure 1: SSA for whole test run

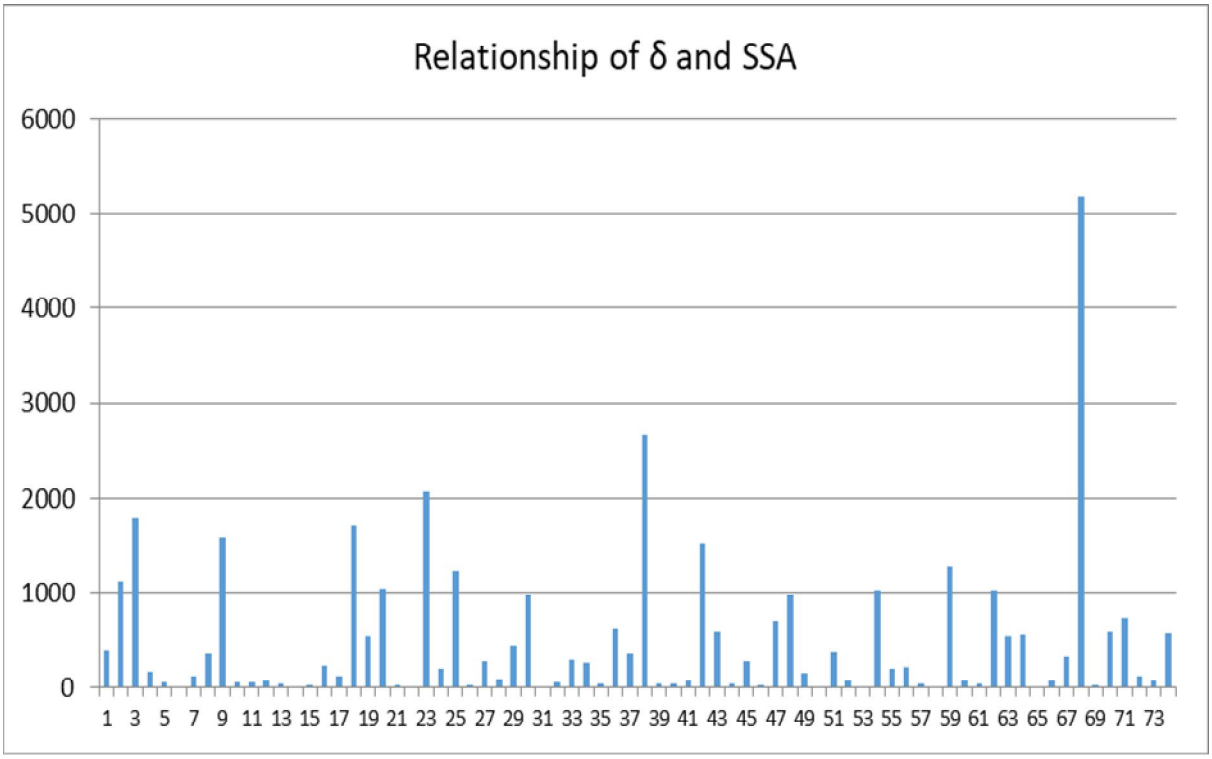

Figure 2: Relationship of δ and SSA

Almost around 100 times we ran with different test cases. The sample data set is shown in the table 2. For each test attempt of the test run the SSA which is defined in equation (1) is found to be between the range (0.00 to 0.021) and the average is 0.005. Gross SSA is found to be between the range (0.060 to 9.068) and the average is 2.07. The value of SSA ensures

the security of the software being developed in the particular module compared to the previous module. Higher the value of SSA will yield higher software security

4. CONCLUSION

The security is one of the most crucial considerations in all software's. This research is initiated due to the inspiration of the biological growth of stem. The result obtained is astonishing. The Investigating the relationship among SSA and δ for security, it is found that almost depends on all the 38 parameters shown in Table 2. δ can be redefined as a function of two independent variables t and m (module difference between the initial and the final module) along the security aspect. This research article shows only the basic statistical analysis, more detailed analyses can be done for further investigation which is out of the scope of this research article.

REFERENCES

1. Andrea Bondavalli, , István Majzik and András Pataricza, **Stochastic Dependability Analysis of System Architecture Based on UML Designs**, Lecture Notes in Computer Science 2677:219-244 · September 2004. https://doi.org/10.1007/3-540-45177-3_10
2. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, **Basic concepts and taxonomy of dependable and secure computing**. IEEE Transactions on Dependable and Secure Computing, 1:11–33, January-March 2004 https://doi.org/10.1109/TDSC.2004.2
3. ISO/IEC 13335: **Information Technology - Guidelines for the management of IT Security**. http://www.iso.org
4. J. Grossman, **Website Vulnerabilities Revealed: What everyone knew, but afraid to believe,** WhiteHat Security, March, 2008
5. S. Shanmuga Priya and P. D. Sheba Keiza Malarchelvi, **Security Deliberations in Software Development Lifecycle**, International Journal of Computer Applications, vol. 975, 2014.
6. Chi-Shiang Cho, Wei-Ho Chung, and Sy-Yen Kuo, **Cyberphysical Security and Dependability Analysis of Digital Control Systems in Nuclear Power Plants**, IEEE Transactions on Systems, Man, and Cybernetics: Systems, Vol. 46(3), MARCH 2016 https://doi.org/10.1109/TSMC.2015.2452897
7. Joe Jarzombek and Karen Mercedes Goertzel, **Security in the Software Life Cycle**, The Journal of Defense Software Engineering, Vol. 19(9), pp 4-9, 2006.
8. Jean-Marc Thiriet, Stéphane Mocanu, **Some Considerations on Dependability Issues and Cyber-Security of Cyber-Physical Systems**, International Conference on Smart Communications in Network Technologies, IEEE, HAL Id: hal-01909025, https://hal.archives-ouvertes.fr/hal-01909025, 2018. https://doi.org/10.1109/SaCoNeT.2018.8585452
9. Hala Assal and Sonia Chiasson, **Security in the Software Development Lifecycle**, Open access to the Proceedings of the Fourteenth Symposium on Usable Privacy and Security is sponsored by USENIX, USA, 2018.
10. Xiaolin Changa, Shaohua Lva, Ricardo J. Rodríguezb and Kishor Trivedic, **Survivability Model for Security and Dependability Analysis of a Vulnerable Critical System**, 27th International Conference on Computer Communication and Networks (ICCCN), IEEE, 2018
11. Yukihiro Chiba and Kichiro Shinozaki, **A Simple Mathematical Model of Growth Pattern in Tree Stem**, Annals of Botany, Elsevier, Vol 73(1), pp 91-98, 1994. https://doi.org/10.1006/anbo.1994.1011
12. M Viswanathan, M Sivaram, D Yuvaraj, AS Mohammed, **Security and privacy protection in cloud computing**, Journal of Advanced Research in Dynamical and Control Systems, 2018, pp.1704-1710
13. Ziyad R. A Ashhab , Mohammed Anbar , Manmeet Mahinderjit Singh , Kamal Alieyan , Wajih I. Abu Ghazaleh, **Detection of HTTP Flooding DDoS Attack using Hadoop with MapReduce : A Survey**, International Journal of Advanced Trends in Computer Science and Engineering, Volume 8, No.1, 2019, pp.71-77.
14. Yogesh Awasthi, RP Agarwal, BK Sharma, **Intellectual Property Right Protection of Browser based Software through Watermarking Technique**, International Journal of Computer Applications, Volume 97– No.12, July 2014, pp.32-36 https://doi.org/10.5120/17061-7468
15. Saravanan, G. Mohammed Gouse, Chiai Mohammed Haji, **Improved Reconfigurable based Lightweight Crypto Algorithms for IoT based Applications**, Journal of Advanced Research in Dynamical and Control Systems, Volume 10, No 12, 2019, pp.186-193.