Layered and hierarchical approach for modelling multidimensional design threats

J.A. Ojeniyi*[1], V.O. Waziri[2], A.M. Aibinu[3] and H.C. Inyiama[4]
[1,2,4]Department of Cyber Security Science, [3]Department of Mechatronics Engineering
Federal University of Technology, Minna, Nigeria
ojeniyija@futminna.edu.ng
+2348073303909
*Corresponding author: ojeniyija@futminna.edu.ng

ABSTRACT
Security of any digitally designed system is not guaranteed until an appropriate modelling and assessment of threat is carried out and proper mitigation is iteratively done. Several existing techniques could not handle the complexities, multi-dimensionality and layered nature inherent in threat modelling of multi-layered systems. This research work sets out to mitigate this particular challenge by designing a simulated computer network multi-layered testbed. Then, hierarchical and layered threat model was developed but iteratively assessed and evaluated to ensure conformity with pre-defined security requirement for the testbed. Multi-level data flow diagram was developed for refined threat model.

Key words: *computer network, testbed, threat model, data flow diagram*

1. BACKGROUND TO THE STUDY

The security of computing systems is not based on assumptions or vendor's claims. Potential threats and vulnerabilities at design-stage and execution time must be put into proper quantitative and qualitative assessment of systems' security requirements. In order to give formal specification of security requirements of computing systems, threat modelling approach is generally used (Myagmar, 2005). The first formal approach to design-level software security modelling was done by the work of (Xu & Nygard, 2005). The properties and inconsistency behaviours between software components were verified in their work. As a result, design-level vulnerabilities were mitigated to a reasonable extent.

The focus of earlier researchers on software-based threat modelling coupled with an increasing rate of storage security breaches necessitated work in other areas. The work of (Hasan, Myagmar, Lee, & Yurcik, 2005) was targeted towards proactive protection of storage systems. Domain-specific modelling approach was used. It is based on two different processes. The first, consideration was given to security principles like confidentiality, integrity, availability and authentication and second, the data lifecycle model was used.
In order to take modelling of threats from design level to execution time, Wang, Wong, & Xu (2007) focused on runtime threat modelling. Unified modelling language sequence diagrams were used to show the consistency of threats at design stage and at runtime. This serves as a guide to code implementation and security testing of such code.

Threat analysis and modelling cannot be limited to qualitative description alone. Quantitative description of security models will help to give discrete measures to system threats. The contribution of (Khan & Hussain, 2010) gives various quantification models that can be used for mathematic or statistical analysis of system security issues. The different research work with their achievement on threat modelling is given in Table 1.

Table 1:Threat Modelling Methodology and Achievements

Author	Methodology	Achievement
(Myagmar, 2005)	Integrated and systematic approach	Modelling of complex systems
(Xu & Nygard, 2005)	Aspect-oriented Petri nets approach	Mitigate design level vulnerabilities
(Hasan et al., 2005)	Domain-specific modelling of storage systems	Proactive protection of storage systems
(Wang et al., 2007)	Design-level and run time-level modelling approach	Mitigates threats metamorphosis during execution
(Khan & Hussain, 2010)	Predictive quantification model	Proactively measures likely threats in quantitative terms
(Gandotra, Singhal, & Bedi, 2012)	Three phased threat-oriented security model	Carries out proactive threat management
(Carlomagno, Martina, Catarina, Price, & Custódio, 2013)	Dynamic and ceremony adjustable model	Mitigates unrealistic model-influenced attacks
(Rostami, Koushanfar, Rajendran, & Karri, 2013)	Hardware-based model for hardware Trojans	Enhanced mitigation of hardware Trojans
(Kaur & Kaur, 2014)	Design-level database threat model	Mitigation of SLQ injection attacks
(Wuyts, Scandariato, & Joosen, 2014)	Privacy-based model	Reduces threats to data privacy and improves correction rate
(Majhi, 2015)	Elliptic curve cryptography and bilinear pairing-based model	Mitigates threats in virtual machine migration auction
(Zawoad, Hasan, & Grimes, 2015)	Trustworthy litigation based model	Improves trustworthiness of cloud storage system

Source: (Zawoad et al., 2015)

2. STATEMENT OF PROBLEM

Existing researches in the literature worked mainly on software-based threats. Little attention was given to hardware-based threats. In addition, there are other dimensions to threat modelling that have not been fully explored such as asset-centric and attacker-centric threats. Essentially, there is persistent problem of threat relations and hierarchy. This problem has caused dependency threat challenges in which one threat depend on the other while another threat is independent of the other. If a relation functions could be defined within various threats categories at different hierarchies, then threat dependency problems will be solved.

3. OBJECTIVE

In mitigating the dependency threat problem, this research work was carried out in order to develop a layered and hierarchical threat model that will consider various nodes (signifying threat elements) at different layers of data flow. Multi-layered data flow diagram was also used to model the flow of data across several layers of developed system.

4. METHODOLOGY

4.1 The Research Design

The methods employed in this research work followed the following stages. The security requirements were first specified and policy formulated as will be required of ideal computer network setup. Based on these requirements, the testbed for the simulation of network scenario was designed and developed. Threat model for validating the security of the developed testbed was then formulated. Iteratively, this model was assessed and evaluated to ascertain its conformity with pre-defined security requirements and policy. Then data flow diagram was used to model different layers of abstraction of the pre-defined security conformed testbed. The research design is depicted by the Fig. 1 and Table 2.

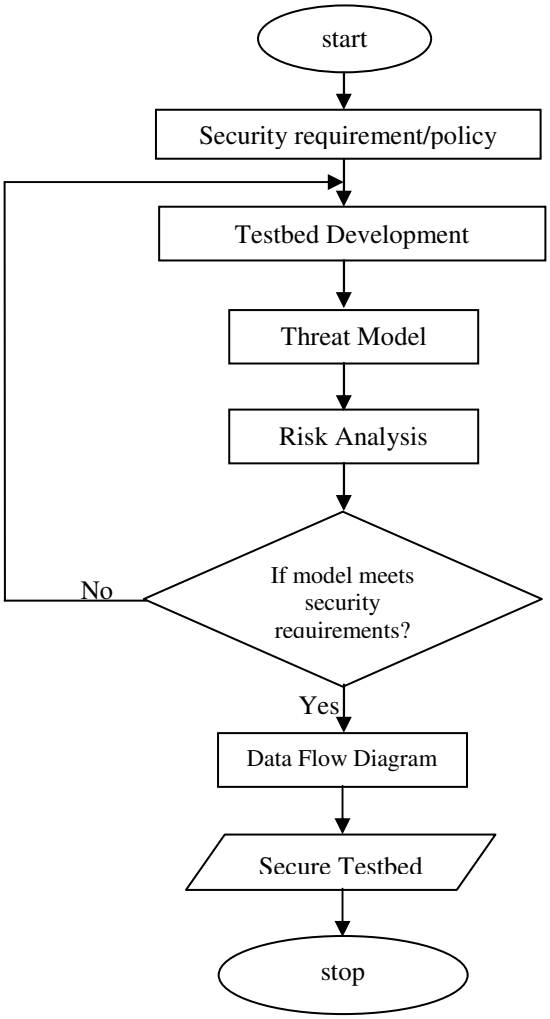

Fig. 1: Layered and hierarchical threat modelling developmental stages

Table 2: Algorithm for layered and hierarchical threat modelling developmental stages

Input: *security requirement and policy*	
Output: *threat model*	
1:	*Security requirement and policy formulation*
2:	*Testbed development*
3:	*Threat modelling*
4:	***While*** *threat model* ***Not*** *security/policy compliant*
5:	***Goto 2:***
6:	***EndWhile***
7:	*Data Flow Diagram*
8:	*Secured Testbed*
9:	*Stop*

4.2 Security Requirement and Policy Formulation

Based on the local network area setup for prototypic implementation of the research design, security hardware like hardware firewalls were used to secure the network from external aggression and to shield the data centre from both internal and external attackers. Trust boundaries were also setup to allow free flow of traffic within the organizational users. Layer 2 and layer 3 authentication of frames and packets were respectively provided by the implementation of hardware-based usage of firewalls. At various internal terminals, user-level authentication and authorization was also ensured through the use of password and biometric features (finger print and facial capturing).

4.3 Testbed Design and Development

VMware Esxi was installed directly on the Hp ProLiant ML 110 G7 server so as to make optimal resource utilization available to the guest machines on the VMware Esxi. Guest machines are the virtual operating systems or software installed on the VMware Esxi among which is Windows 8.1. In order to set up the testbed, Graphical Network Simulator 3 (GNS3) and Virtual box were installed on the Windows 8.1. The GNS3 was used in this research work to set up the testbed as shown in Figure 2. The GNS3 simulated testbed contains the following network devices: a router which was labelled as Edge-Router, two firewalls labelled as Firewall1 and Firewall2, layer 2 switch labelled as SW1 and layer 3 switch labelled as cloud1. The live Internet Operating Systems (IOS) of these devices are contained in the QEMU and dynamips of GNS3. As clearly labelled in Figure 2, the network terminals are ADMIN_PC, Workstation1, Workstation2, Database Server and Web/portal Server. The live operating systems for the network terminals were safely housed in virtual box as shown in Figure 3. VMware Esxi cannot be accessed directly on the server but through a remote client system. Four systems were used as remote login systems to the VMware Esxi on the server.

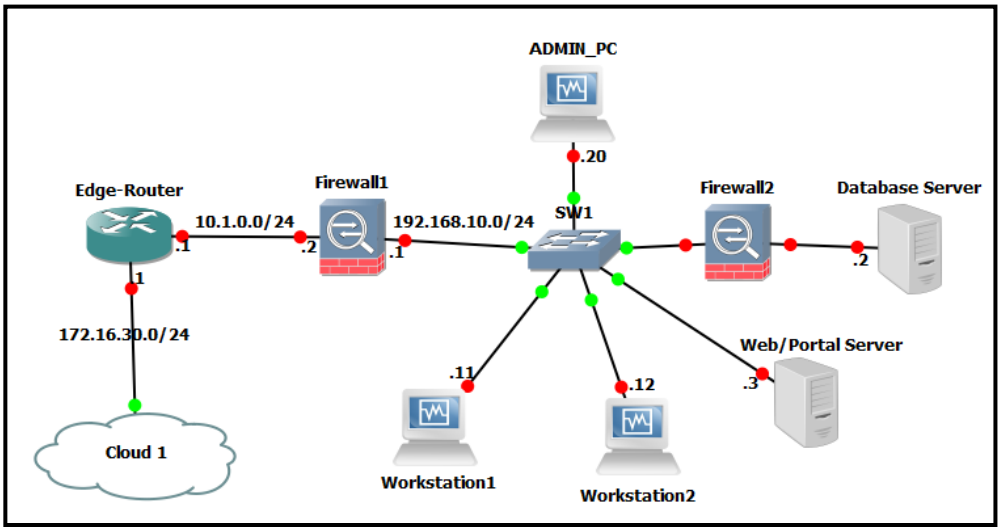

Fig. 2: GNS3 simulated testbed

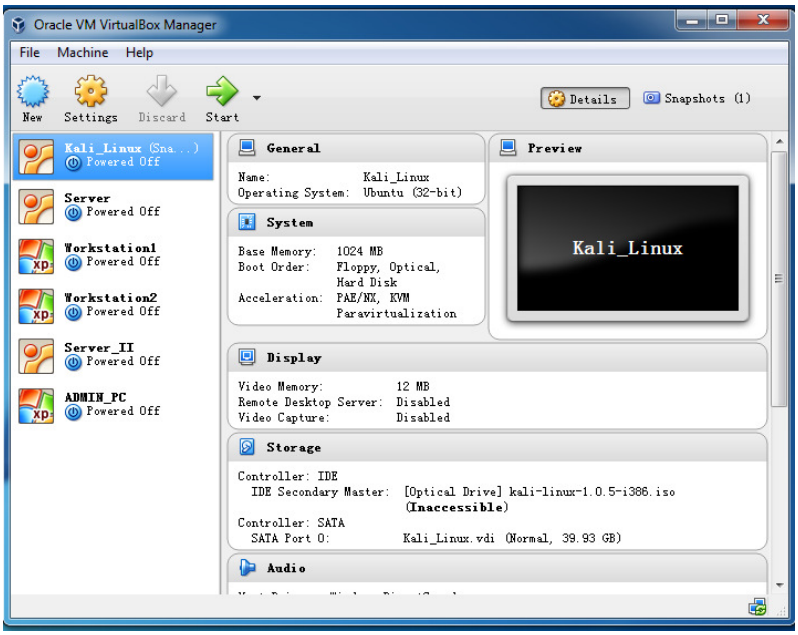

Fig. 3: Virtual box containing operating systems of terminals in GNS3 simulated testbed

5. RESULT PRESENTATION

5.1 Threat Model definition

Definition 1: The multi-layered threat model denoted by M is a 3-tuple (L, AG, H) as shown in (1), where L is a set of hierarchical layered threats, AG is a set of threat assessment algorithms, H is a set of historical statistical information.

$$M = (L, AG, H) \tag{1}$$

Definition 2: A layered threat denoted by L_i is a 3-tuple (T_C, N, R) for $i=1,2,\ldots n$ as shown in (2), where T_C, as shown in (3), is a set of threat categories consisting of software centric threats, T_{SW}, asset centric threats, T_{AS} and attacker centric threats, T_{AK}, N is a set of nodes in L_i, R is a set of relations denoting possible connections or communications between nodes.

$$L_i = (T_C, N, R) \tag{2}$$

$$T_C = \{T_{SW}, T_{AS}, T_{AK}\} \tag{3}$$

Definition 3: N is a set of nodes in a layered threat, consisting of node elements N_i, $i = 1, 2, \ldots, n$. A node N_i consists of location attribute denoted by A_L, time attribute denoted by A_T, semantic and context attribute denoted by A_{SC}, logic attribute denoted by A_{LG}, as shown in (4).

$$N = \{N_i \mid A_L, A_T, A_{SC}, A_{LG}\} \tag{4}$$

Definition 4: A location attribute, A_L, as shown in (5), is a set consisting of hardware node attribute denoted by L_{HW}, hypervisor node attribute denoted by L_{HP}, operating system node attribute denoted by L_{OS}, gns3 node attribute denoted by L_{GN}, and gns3 network node attribute denoted by L_{NT}.

$$A_L = \{L_{HW}, L_{HP}, L_{OS}, L_{GN}, L_{NT}\} \tag{5}$$

Then, let N^{HW} denotes a set of hardware nodes, N^{HP} denotes a set of hypervisor nodes, N^{OS} denotes a set of operating system nodes, N^{GN} denotes a set of gns3 nodes and N^{NT} denotes a set of gns3 network nodes.

Definition 5: A_T is a set of time attributes of nodes. The lower boundary of this attribute is estimated from the time, t_L taken by light travel.

$$A_T \in [t_L, +\infty) \tag{6}$$

Definition 6: As shown in (7), A_{SC} is a set of semantic and context attribute of nodes, consisting of preconditions of a threat goal denoted by C_{pre}, postconditions of a threat goal denoted by C_{post}, gns3 deployment information denoted by G_{DE}, gns3 runtime information denoted by G_{RU}, threat category, T_C as in (3) and attack scenario dented by S_A. C_{pre}, C_{post}, G_{DE}, G_{RU}, T_C are semantic attributes while S_A is a contextual attribute.

$$A_{SC} = \{C_{pre}, C_{post}, G_{DE}, G_{RU}, T_C, S_A\} \tag{7}$$

In definition 6, the preconditions include assumptions for an attack, about the attacker or the states of the software that are necessary for an attack to succeed, such as resources, access, skills or knowledge that the attacker must possess, and the level of risk that the attacker must be willing to bear. The postconditions include knowledge acquired by the attacker and changes to the software states that result from successfully implementing the attack steps when the preconditions hold, such as system paralysis, system performs malicious function and to be controlled by attacker. GNS3 deployment information contains security information in deployment, such as secure target in deployment and methods of managing secure function and so on. GNS3 runtime information contains information of environment in which the software gns3 may be applied, such as description of runtime parameters, runtime status, runtime security states, and so on. These GNS3 deployment and runtime information are used for researching on the relationship among software threats or software centric threats. They are used to enhance secure deployment and applications. Attack scenario is the situation of carrying out an attack. Security experts can obtain a lot of attack scenario information from attack case and historical statistical information.

Definition 7: As shown in (8) A_{LG} is a set of logic type attributes consisting of A_{LG}^{AND} which denotes a AND node attribute, A_{LG}^{OR} denotes OR node attribute. Let N^{AND} denotes a set of AND nodes, N^{OR} denotes a set of OR nodes.

$$A_{LG} = \left\{A_{LG}^{AND}, A_{LG}^{OR}\right\} \tag{8}$$

In definition 7, AND node signifies a type of node that can only achieve its threat goal provided all the threat goals of its sub-nodes are achieved. OR node represents a type of node that can achieve its threat goal when the threat goal of any of its sub-nodes is achieved.

Definition 8: *R* is a set of relations between nodes as in (9)

$$R = \left\{R_{us} \mid u, s \in N\right\} \tag{9}$$

Where relation R_{us} is a 6 – tuple as shown in (10)

$$R_{us} = \left\langle u, s, E_q, C_O, P, D_a \right\rangle \tag{10}$$

Where *u* and *s* denote the upper node and sub node respectively. E_q denotes the requirement of special equipment or not, $E_q \in \{true, false\}$. C_O denotes attack cost, $C_O \in [0,+\infty)$. *P* denotes the probability of carrying out an attack, $P \in [0,1]$. D_a denotes attack damage. D_a is directly proportional to P and inversely proportional to C_O, resulting to (11).

$$D_a = \frac{p}{C_O} \tag{11}$$

Definition 9: Let P_{AT} denotes a set of attack path, $P_{AT}[n]$ denotes n^{th} attack path. An attack path is a minimum cut set of *L*. A *node set* is a cut set, if: (i) it is a *terminal node* set of *L*; and (ii) if all the threat goals of these terminal nodes are achieved, the final threat goal of the *initial node* in *L* can be achieved. Let C_U denotes a cut set. A node is a *minimum cut set*, if: (i) it is cut set; and (ii) if any terminal node is removed from the cut set, the cut set is not a cut set anymore. Let $C_{U_{\min}}$ denotes a minimum cut set.

Definition 10: As shown in (12), *AG* is a set of threat assessment algorithms consisting AG_{MC} which denotes model constructing algorithm, AG_{AT} denotes attack path detection algorithm, AG_{HL} denotes hierarchical layered threat model based threat assessment algorithm.

$$AG = \left\{AG_{MC}, AG_{AT}, AG_{HL}\right\} \tag{12}$$

Definition 11: *H* is a set of historical statistical information. This information is a source of help in designing mitigation strategies. As shown in (13), the set *H* consists of history of threat category denoted by HT_C, history of nodes denoted by HN_i, history of relation denoted by *HR*, history of algorithms denoted by *HAG*.

$$H = \left\{HT_C, HN_i, HR, HAG\right\} \tag{13}$$

In definition 11, HT_C consists of history of software centric threat denoted by hT_{SW}, history of asset centric threat denoted by hT_{AS}, history of attacker centric threat denoted by hT_{AK} as shown in (14).

$$HT_C = \left\{hT_{SW}, hT_{AS}, hT_{AK}\right\} \tag{14}$$

Similarly, the histories of nodes, relations and algorithms are respectively shown in (15), (17) and (19). The histories of node attributes are shown in (16) while the histories of relation quantifiers between nodes u and s are shown in (18).

$$HN = \left\{hN_i, i = 1,2,...,n\right\} \tag{15}$$

$$hAN_i = \left\{A_L, A_T, A_{SC}, A_{LG}\right\} \tag{16}$$

$$HR = \left\{hR_{us} \mid u, s \in N\right\} \tag{17}$$

$$hR_{us} = \left\langle hu, hs, hE_q, hC_O, hP, hD_a \right\rangle \tag{18}$$

$$HAG = \left\{hAG_{MC}, hAG_{AT}, hAG_{HL}\right\} \tag{19}$$

Definition 12: For quantitative analysis of the element histories of (14), hT_{SW} consists of probability of vulnerability denoted by $P_V(t)$ and probability of threat denoted by $P_T(t)$ where t signifies time t, hT_{AS} consists of probability of risk, $P_R(t)$ and asset reliability denoted by by $R_{EL}(t)$, hT_{AK} consists of probability of an attack denoted by $P_A(t)$ and consequence of an attack, $C(t)$.

$$P_T(t) = \frac{T(t)}{T} \tag{20}$$

$$P_V(t) = \frac{V(t)}{V} \tag{3.21}$$

When the occurrence of threats and vulnerabilities are not independent, then the equations (20) and (21) translates into (22) and (23).

$$P_{T/V}(t) = \frac{p_{V \cap T}(t)}{P_V(t)} \tag{22}$$

$$P_{V/T}(t) = \frac{P_{T \cup V}(t)}{P_T(t)} \tag{23}$$

If the reliability of the asset is known, then equation (20) can be computed as in (24).

$$P_T(t) = P_V(t) * R_{EL}(t) \tag{24}$$

$$R_{EL}(t) = e^{-ft} \tag{25}$$

Where f is given as the failure rate.

$$P_R(t) = P_V(t) * P_T(t) * C(t) \tag{26}$$

$$P_A(t) = e^{-\lambda t} * \frac{(\lambda t)^n}{n!} \tag{27}$$

Where λ is the inter-arrival rate of attacks and n is the number of attacks occurred in time interval t.

$$C(t) = \frac{P_A(t)}{P_T(t)} \tag{28}$$

5.2 Threat assessment algorithms

Threat assessment algorithm is the algorithm that is used to generate the model of the threats or vulnerabilities inherent in your developed testbed based on layers and hierarchy. The algorithm is shown in Table 3.

Table 3: Algorithm for hierarchical layered threat model construction

***Input**: node set* $N = \{N_i \mid A_L, A_T, A_{SC}, A_{LG}\}$, *relation set* $R = \{R_{IJ} \mid i, j \in N\}$	
***Output**: multi-layered threat model L*	
1:	*Initialize L*
2:	*get a node* N_u
3:	***While*** $N_u \notin N'$ *Do*
4:	$t \leftarrow N_u$;
5:	$N_i \leftarrow R_{u,s}, s \in N$
6:	***EndWhile***
7:	$L \leftarrow N_u$
8:	*Repeat until* $N \neq \phi$
9:	*Return L*

5.3 Threat model data flow diagram for simulating testbed

The threat models for simulating testbed particularly showing the processes and threats involved are shown according to level of abstraction from Figure 4 to 8.

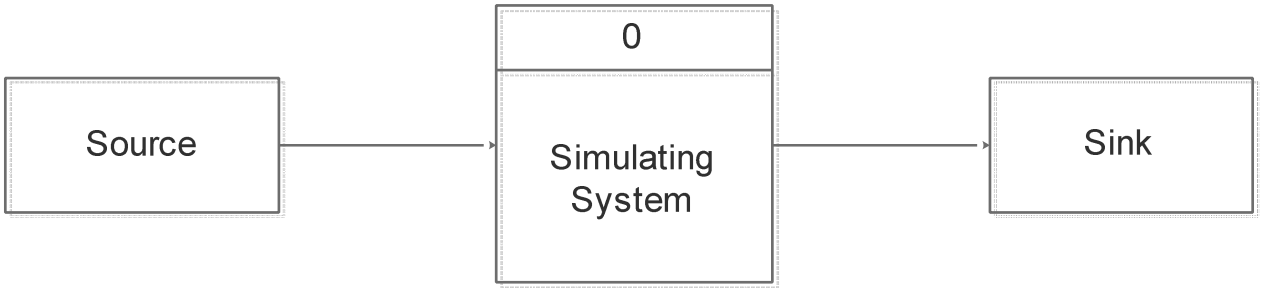

Figure 4: Context Data Flow Diagram for Simulating Testbed

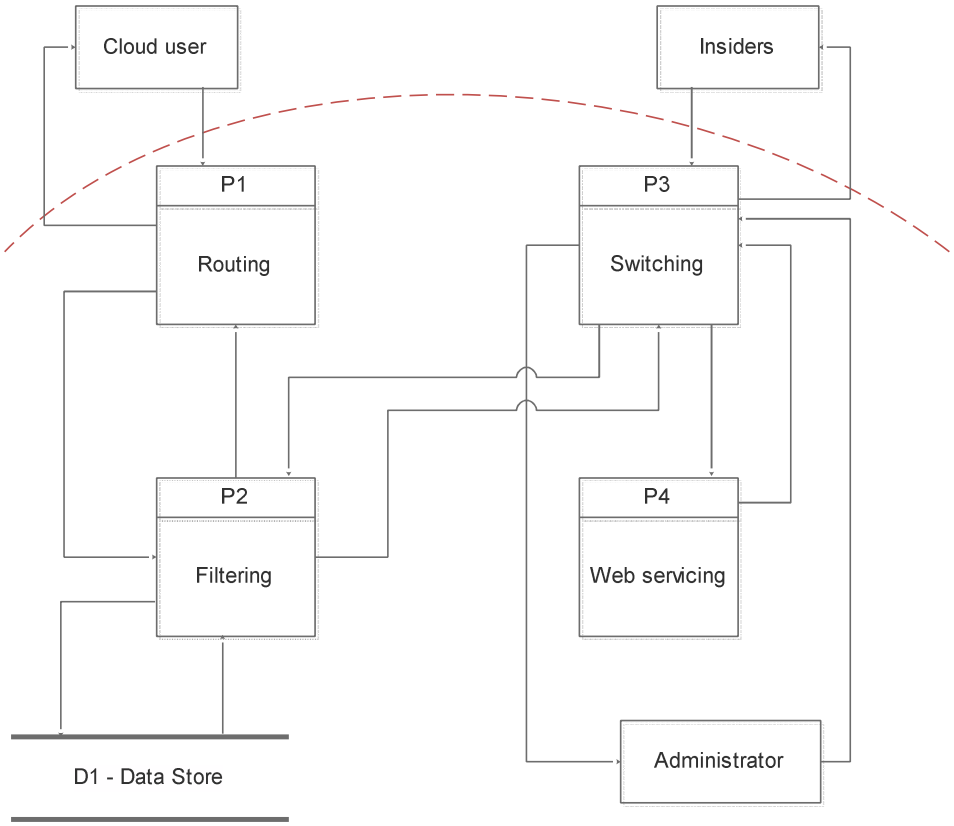

Fig. 5: Level-0 Data Flow Diagram

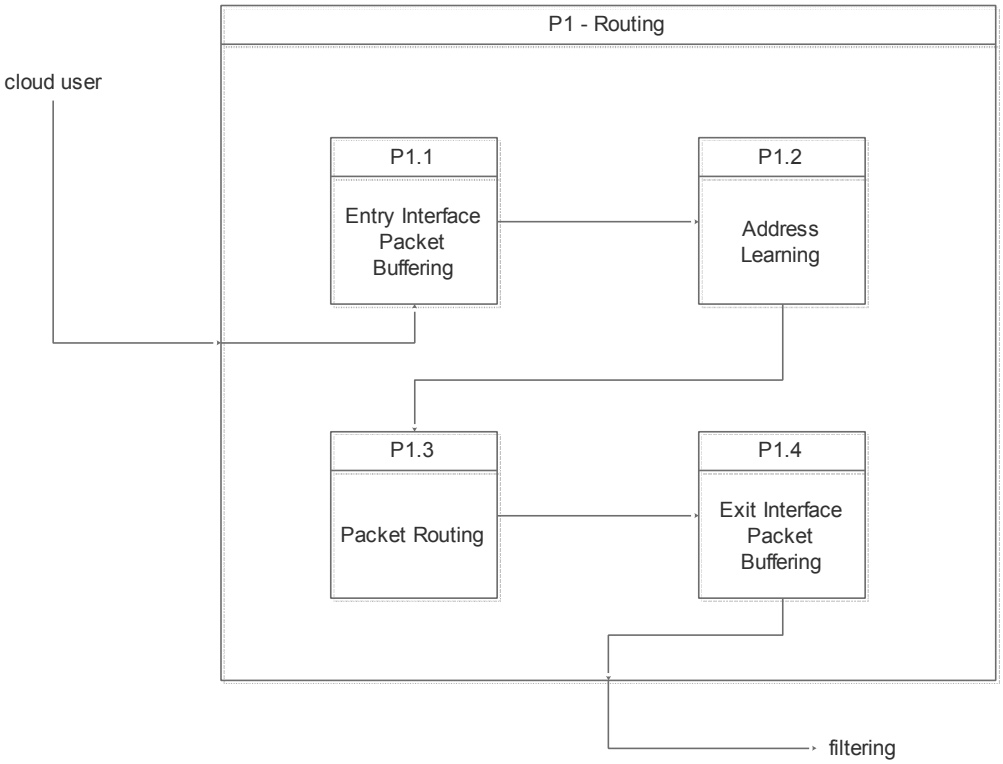

Fig. 6: Level-1 Routing Data Flow Diagrams for sub-processes in P1

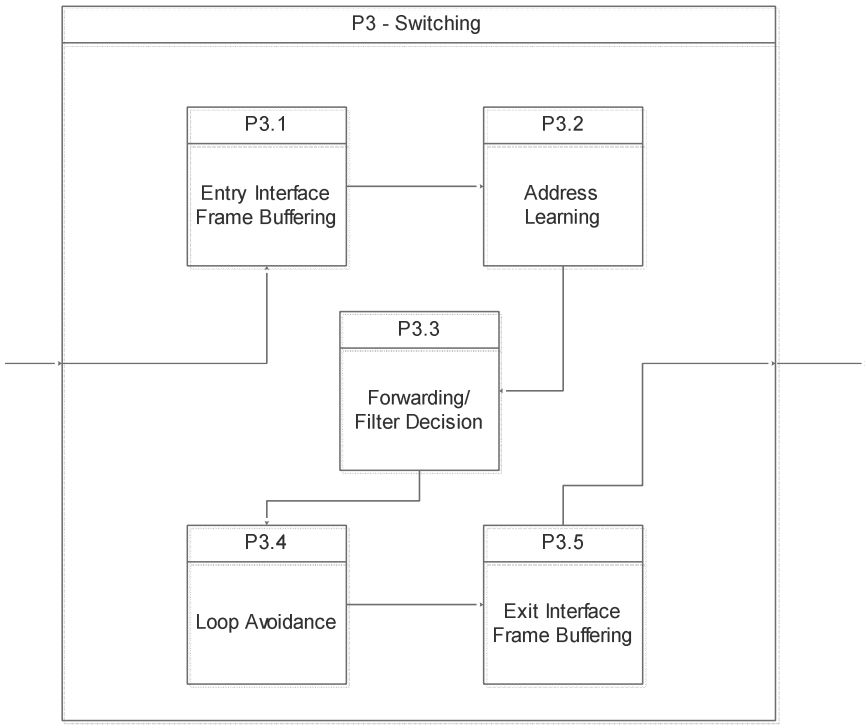

Fig. 7: Level-1 Switching Data Flow Diagrams for sub-processes in P3

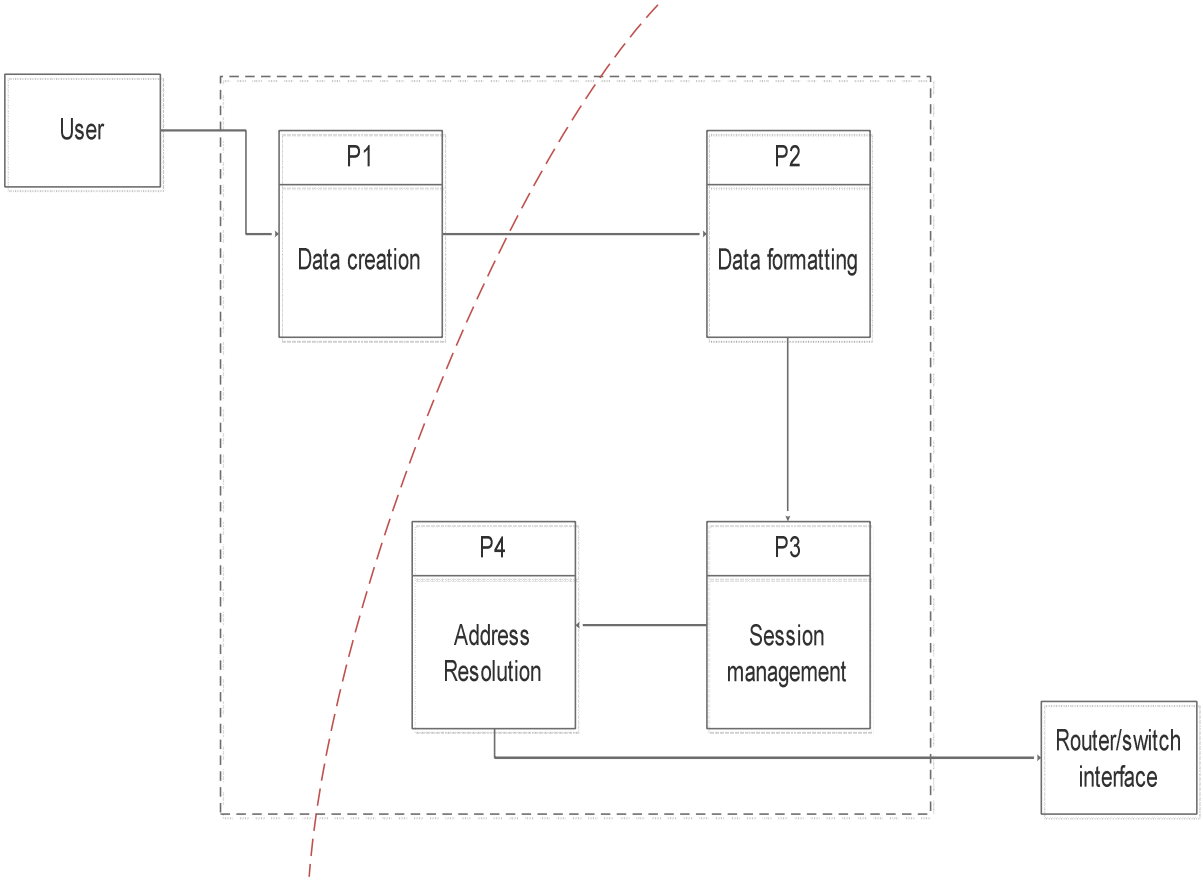

Fig. 8: Levcl-1 cloud/insider user data flow diagram

6. DISCUSSION OF FINDINGS

In equation (1), the 3-tuple basic threat model equation was formulated whose meaning was expanded in the subsequent definitions. Particularly, equation (2) consists of the three major threat categories. This enhances the inclusiveness of the threat model. The threat assessment algorithm was meant to track various nodes of data communication across hierarchical layer. In order to properly model the movement of data across these nodes, layered data flow diagrams were also used to carry out the modelling. In Fig. 4, the context diagram shows the very high level of the developed testbed without giving the details. Fig. 5 shows the level-0 data flow diagram. It is the high level diagram that describes data flow in an individual process. The low level detailed sub-components flow is shown and described in Fig. 6.

7. CONCLUDING REMARKS

This particular technique to threat modelling of computer network has provided layered and hierarchical-based approach which is encompassing. Three major threat categories were taken into consideration. Even though the mathematical-based model and data flow diagram were developed it is recommended that further assessment and validation be carried out on the model and the diagram.

8. CONTRIBUTIONS TO KNOWLEDGE

The major contribution of this research work is the formulation and development of a layered and hierarchical threat model for validating the security of layered-based testbed model.

REFERENCES

Carlomagno, M., Martina, J. E., Catarina, S., Price, G., & Custódio, R. F. (2013). An Updated Threat Model for Security Ceremonies. In *Symposium on Applied Computing* (pp. 1836–1843).

Gandotra, V., Singhal, A., & Bedi, P. (2012). Threat-Oriented Security Framework: A Proactive Approach in Threat Management. *Procedia Technology*, *4*(2012), 487–494. http://doi.org/10.1016/j.protcy.2012.05.078

Hasan, R., Myagmar, S., Lee, A. J., & Yurcik, W. (2005). Toward a threat model for storage systems. In V. Atluri (Ed.), *Proceedings of the 2005 ACM workshop on Storage security and survivability - StorageSS '05* (p. 94). Alexandria, VA, USA: ACM New York, NY, USA. http://doi.org/10.1145/1103780.1103795

Kaur, N., & Kaur, P. (2014). Mitigation of SQL Injection Attacks using Threat Modeling. *ACM SIGSOFT Software Engineering Notes*, *39*(6), 8–13. http://doi.org/10.1145/2674632.2674638

Khan, M. A., & Hussain, M. (2010). Cyber Security Quantification Model. *Bahria University Journal of Information Amp; Communication Technologies VO - 3*, (1), 39. http://doi.org/10.1145/1854099.1854130

Majhi, S. K. (2015). A Security Enforcement Framework for Virtual Machine Migration Auction Categories and Subject Descriptors. In E. Al-Shaer (Ed.), *Proceedings of the 2015 Workshop on Automated Decision Making for Active Cyber Defense* (pp. 47–53). Denver, CO, USA: ACM New York, NY, USA.

Myagmar, S. (2005). Threat Modeling as a Basis for Security Requirements. In V. Atluri (Ed.), *In StorageSS '05: Proceedings of the 2005 ACM workshop on Storage security and survivability* (pp. 94–102).

Rostami, M., Koushanfar, F., Rajendran, J., & Karri, R. (2013). Hardware security: Threat models and metrics. In K. Embler (Ed.), *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (pp. 819–823). Hilton San Jose 300 Almaden Blvd. San Jose, CA, USA: IEEE Council on Electronic Design Automation. http://doi.org/10.1109/ICCAD.2013.6691207

Wang, L., Wong, E., & Xu, D. (2007). A threat model driven approach for security testing. In *Proceedings of the Third International Workshop on Software Engineering for Secure Systems* (p. 10). http://doi.org/10.1109/SESS.2007.2

Wuyts, K., Scandariato, R., & Joosen, W. (2014). Empirical evaluation of a privacy-focused threat modeling methodology. *Journal of Systems and Software*, *96*, 122–138. http://doi.org/10.1016/j.jss.2014.05.075

Xu, D., & Nygard, K. (2005). A threat-driven approach to modeling and verifying secure software. In D. Redmiles (Ed.), *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering - ASE '05* (p. 342). Long Beach, CA, USA: ACM New York, NY, USA. http://doi.org/10.1145/1101908.1101965

Zawoad, S., Hasan, R., & Grimes, J. (2015). LINCS: Towards building a trustworthy litigation hold enabled cloud storage system. *Digital Investigation*, *14*(2015), S55–S67. http://doi.org/10.1016/j.diin.2015.05.014