AENSI Journals

**Australian Journal of Basic and Applied Sciences**

Journal home page: www.ajbasweb.com

# Improving Intrusion Detection System Using Spiking Neural P System: An Appraisal

**Rufai Kazeem Idowu, Ravie Chandren a/l Muniyandi and Zulaiha Ali Othman**

*School of Computer Science, Faculty of Information Science & Technology, Universiti Kebangsaan Malaysia*

ARTICLE INFO

***Article history:***
*Received 4 September 2013*
*Received in revised form 21 October 2013*
*Accepted 29October 2013*
*Available online 18 November 2013*

***Key words:***
*Spiking Neural P System*
*Intrusion Detection System*
*Parallel Computation*
*Membrane Computing*

ABSTRACT

'Intrusion' has been acknowledged as a hydra-headed monster which has been around since the debut of the computer machine. In an attempt to curtail or reduce this menace to the barest level, various approaches have been introduced in the past. However, these intrusion detection mechanisms have the main problems of low detection and high false alarm rates among others. In this work, the class of membrane systems which is inspired by the spiking of biological neurons, known as Spiking Neural P systems (simply called SN P systems), is investigated. This SNP system, is similar to other P system variants (Tissue-like and Cell-like), which are parallel, nondeterministic, and distributed computing devices. Succinctly therefore, this paper makes a preview of Intrusion Detection Systems and Spiking Neural P system with a view to establishing a link between them. Effort is also made in x-raying a neuron, application of SN P rules and its computability. Finally, some of the benefits of SN P system which could be harnessed to tackle the challenges of a typical IDS are also highlighted.



## INTRODUCTION

Attempts to breach information security are rising every day, despite the availability of the vulnerability assessment tools. This is not unconnected with the fact that some mischief makers who are laddened with evil tendencies constantly finding ways of compromising the integrity, confidentiality and availability of the electronic information systems through intrusion. Intrusion detection System (Northcutt, S. and J. Novak, 2002).

Therefore, a system which is said to be intruded suffers from any of the following three attacks; a) External attacks which comprise of password cracking, network sniffing, machine and services discovery utilities, packet spoofing and flooding utilities. b) Internal penetrations, made up of masqueraders, clandestine users. c) Misfeasors which constitute authorized misuse. Therefore, in an attempt by concerned individuals and corporate bodies to find lasting solution to this intrusion imbroglio, different approaches have been introduced. Regrettably however, none of the intrusion–detection approaches so far discovered is sufficient in its entirety to address all likely threat a computer system may encounter. From another angle, a school of thought believes that a computer system's primary defence/access control mechanisms should be sufficient to curtail this issue of intrusion, but this has likewise proved furtile.

It is in the light of the above that attention is now being shifted to an aspect of membrane computing-Spiking Neural P System to determine how suitable it could be in resolving some of the challenges imposed by the destroyer called '*intrusion'*.

This paper is arranged into the following sections: Section 2 briefly discusses IDS, its basic requirements and Taxonomy are highlighted. Also, while effort is made is distinguishing an Anomaly based IDS from Signature based IDS, the challenges confronting today's IDS are likewise not left out. Section 3 presents the x-ray of a typical neuron as well as the SN P system's rules. Specifically, sections 4 and 5 dwell on the computation (with relevant example), model and simulation of SN P system. The sixth section focuses on the positive impacts of SN P system on IDS. The final section, section 7 concludes the paper through summary.

### *Intrusion Detection System:*

### *The Concept of Intrusion:*

Intuitively, intrusion in an information system is an activity which deliberately violates the security policy of that system. An intrusion is any set of action which attempts to compromise the integrity, confidence or availability of resource. Simply put, an intrusion is a security threat deliberately done to access and/or manipulate information and to render a system unreliable or unusable.

**Corresponding Author:** Rufai Kazeem Idowu, School of Computer Science, Universiti Kebangsaan Malaysia, 43600, Bangi, Selangor, Malaysia.
E-mail: ruffyk2001@yahoo.com

Australian Journal of Basic and Applied Sciences, 7(11) Sep 2013, Pages: 357-365

Researches have shown that computer systems suffer from security vulnerabilities (like intrusion) regardless of their purpose, manufacturer, or origin, and that it is both technically difficult and economically costly to build and maintain computer systems and networks which are not susceptible to attacks.

If an intrusion is a security threat which is deliberately done to access and/or manipulate information and to render a system unreliable or unusable, then an IDS is a device which monitors it. It checks a network for potentially malicious activity and reports it to administrators for further investigation. IDSs are a critical component of any security infrastructure. Also, an Intrusion Detection System analyzes information from a computer or a network to detect malicious actions and behaviors that can compromise the security of a computer system (Bace, R.G., 2000). Therefore, it is a software product of hardware technology that automate a monitoring process of events which occur in a computer system or network with a view to analysing them for signs of intrusion.

In similar perspective, Debar *et al* (1999) submitted that an IDS is a system which dynamically monitors the action taken in a given environment, and decides whether or not these actions are symptomatic of an attack or constitute a legitimate use of the environment. They concluded by presenting the organization of a generalized IDS in the fig. 1 below. Solid lines indicate data/control flow, while dashed lines indicate responses to intrusive activities.

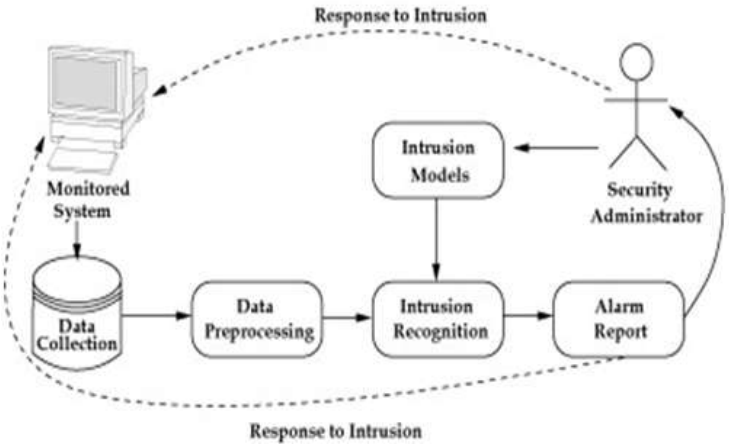


**Fig. 1:** Organization of an IDS.

***Requirements and Taxonomy of Intrusion Detection System:***

The basic requirements of an ideal IDS include:

- It must be *robust* in such a way that the detection points should be more than one which are resilient to attack.
- It should be able to *configure* itself easily to the local requirements of each host or each network component.
- It should be easy to *extend* the scope of IDS monitoring in a simple manner regardless of operating systems.
- S*calability* must be attainable so as to gather and analyse high-volume of audit data correctly from distributed hosts.
- It should be able to *adjust* constantly to the dynamism of network attacks.
- It should be *efficient* in a way that it is simple and lightweight enough to impose a low overhead on the monitored host.

***Anomaly Based IDS versus Signature Based IDS:***

Intrusion detection techniques may be grouped into two classes based on how they monitor and detect attacks. These groupings: *anomaly based detection* and *signature based (misuse) detection*.

An ***Anomaly***-Based Intrusion Detection System is a system for detecting computer intrusions and misuse by monitoring system's activities and classifying it as either *normal* or *anomalous*. It is made up of a complete set of valid requests which are identified accurately; hence it is possible to detect new attacks and variations of attacks. The classification is based on heuristics or rules, rather than patterns or signatures, and attempts to detect any type of misuse that falls out of normal system operation. This is as opposed to signature based systems which can only detect attacks for which a signature has previously been created. In order to determine what attack traffic is, the system must be taught to recognize normal system activity. This can be accomplished in several ways, most often with artificial intelligence type techniques. Systems using neural networks have been used to a very great extent. Another method is to define what normal usage of the system comprises using

Australian Journal of Basic and Applied Sciences, 7(11) Sep 2013, Pages: 357-365

a strict mathematical model, and flag any deviation from this as an attack. This is known as strict anomaly detection (Torrano, C., 2010). Other advantages include: (a) the normal behavior is defined, it is not needed to define a signature for every attack and their variations. (b) It is scalable (c) Resources usage is minimal.

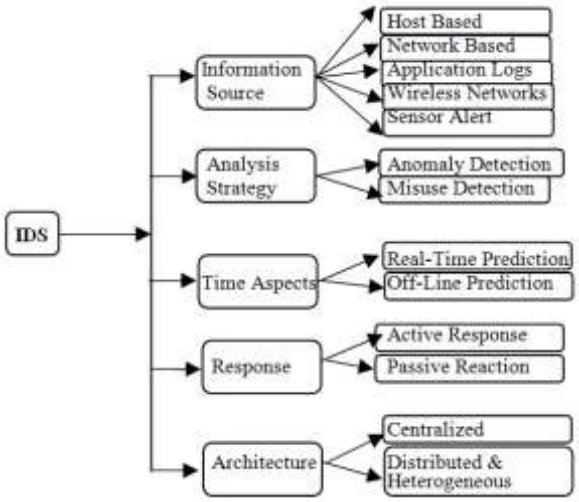


**Fig. 2:** Taxonomy of Intrusion Detection Systems.

However, the major problem of all anomaly detection approaches is that the subject's normal behavior is modeled on the basis of the (audit) data collected over a period of normal operation. If undiscovered, intrusive activities occurring during this period, would be considered normal activities.

A ***Signature*** based IDS monitors packets in a network and compares with pre-configured and pre-determined attack patterns known as signatures. The issue here is that there will be lag between the new threat discovered and Signature being applied in IDS for detecting the threat. During this lag time an IDS will be unable to identify threat. Signature engines also have another disadvantage. Because they only detect known attacks, a signature must be created for every attack, and novel attacks cannot be detected. Misuse detection catches intrusions in terms of the characteristics of known attacks or system vulnerabilities; any action that conforms to the pattern of a known attack or vulnerability is considered intrusive. The rationale of misuse detection is that with additional knowledge of known attacks or vulnerabilities, we can potentially detect these attacks more precisely and more quickly.

Other merits of signature based IDS are: (a) Signatures are easy to develop and understand if the behavior to be identified is known (b) A signature has to be defined for every attack and their variations (c) High usage of resources

The major problems in misuse detection are the representation of known attack patterns and the difficulty of detecting new attacks. This therefore implies that they would not be to identify truly new attacks since no pattern may be provided for them.

***Challenges of Today's IDS:***

Based on the degree of sophistication in the contemporary age, IDS architectures are becoming grossly inadequate to meet up with the highly complex information infrastructure on ground (Garcia-Teodoro, P., 2009). In the configuration of a typical IDS, human security's expertise is usually required for translation into signatures known as patterns or set of rules so as to measure variables being collected. This approach is however not without its drawbacks especially as it relates to:

(i) **True Positive (TP):** A legitimate attack which triggers an IDS to produce an alarm.
(ii) **False Positive (FP):** An event signaling an IDS to produce an alarm when no attack has taken place.
(iii) **False Negative (FN):** A failure of an IDS to detect an actual attack.
(iv) **True Negative (TN):** When no attack has taken place and no alarm is raised.

Among others, these challenges include:

- *Low Attack Detection Rate (ADR)*: This is defined as the number of attack instances detected by the system (True Positive) divided by the total number of attack instances present in the test set. So, a good IDS should be able to return high ADR.

$$ADR = \frac{TP}{TP + FP} \quad (1)$$

Australian Journal of Basic and Applied Sciences, 7(11) Sep 2013, Pages: 357-365

- *High False Alarms Rate (FAR):* Is defined as the number of 'normal' patterns classified as attacks (False Positive) divided by the total number of 'normal' patterns. Actually, the Internet is a very noisy environment, a poor IDS would therefore be confused in generating unwanted alarm.

$$FAR = \frac{FP}{FP + TN} \quad (2)$$

- *Poor Classification Accuracy Rate (CAR):* This is defined as the portion of True Positives (TP) and True Negatives (TN) in the population of all instances. TP is a legitimate attack which triggers an IDS to produce an alarm while TN is when no attack has taken place and no alarm is raised. So, a good IDS should normally be able to classify an attack as intrusion and vice versa.

$$CAR = \frac{TP + TN}{TP + TN + FP + FN} \quad (3)$$

- *No trustworthy automatic response*: It is just not sufficient for an IDS to monitor a network, garner logs and make a report; it should be able to put the attack on hold. The truth of the matter is that as at now most detection mechanisms we have are mere 'reporters' because they wait for human intervention before responding.

***Spiking Neural P System:***

Membrane computing is one of the recent branches of natural computing. Based on how cells are organized in different computing models, MC is mainly classified into cell-like P systems, tissue-like P systems and neural-like P systems. The SN P therefore is a class of neural-like P system.

Spiking neural P systems are a class of distributed and parallel computing models inspired by the neurophysiological behaviour of neurons sending electrical impulses (spikes) along axons to other neurons. An SN P system consists of a set of neurons placed in the nodes of a directed graph, where neurons send signals (spikes) along synapses. A neuron can send a spike to all neurons connected by an outgoing synapse from the neuron by using the spiking rule, and the neuron can also remove its spikes contained by using the forgetting rules. One of the neuron is considered to be the output neuron, and its spikes are also sent to the environment. An SN P system works in the following way. A global clock is assumed and in each time unit each neuron which can use a rule should do it (the system is synchronized), but the work of the system is sequential locally: only (at most) one rule is used in each neuron

The Spiking Neural P System (usually abbreviated as SN P systems) is an off-shoot of Membrane computing. SN P system has a biological motivation based on recent discoveries on neural coding. In all, SN P systems are a class of distributed and parallel computing models inspired by the neurophysiological behaviour of neurons sending electrical impulses (spikes) along axons to other neurons. In particular, SNP systems try to give a modest but formal representation of a special type of cell known as the neuron and their interactions with one another.

An SN P system consists of a set of neurons placed in the nodes of a directed graph, where neurons send signals (spikes) along synapses. A neuron can send a spike to all neurons connected by an outgoing synapse from the neuron by using the spiking rule, and the neuron can also remove its spikes contained by using the forgetting rules. One of the neuron is considered to be the output neuron, and its spikes are also sent to the environment. An SN P system works in the following way. A global clock is assumed and in each time unit each neuron which can use a rule should do it (the system is synchronized), but the work of the system is sequential locally: only (at most) one rule is used in each neuron.

The only one type of object, called the 'spike' and the main information one works with is the distance between consecutive spikes.

***X-Raying a Neuron:***

SN P system consists of a set of neurons (cells, consisting of only one membrane) placed in the nodes of a graph and sending signals (spikes) along synapses (edges of the graph), under the control of firing rules.

The elementary processing units in the nervous system (as it relates to Biology) are called *neurons* which are connected to each other in an intricate pattern. In all, neurons can be of different sizes and shapes. A typical neuron has three functionally distinct parts called ***dendrites, soma*** and ***axon***, as captured in diagram (Fig 2) below.

Dendrites serve as input device because they collect signals from other neurons and transmit them to the soma. On the other hand, the soma acts as a processing unit-CPU- in the sense that it performs an important non-linear processing step thereby manipulating the received signal. Finally, the axon is the output device which delivers the signals to other neurons. Also, in SN P systems, the axon is not just a simple transmitter of spikes, but communicates by so doing.

Australian Journal of Basic and Applied Sciences, 7(11) Sep 2013, Pages: 357-365

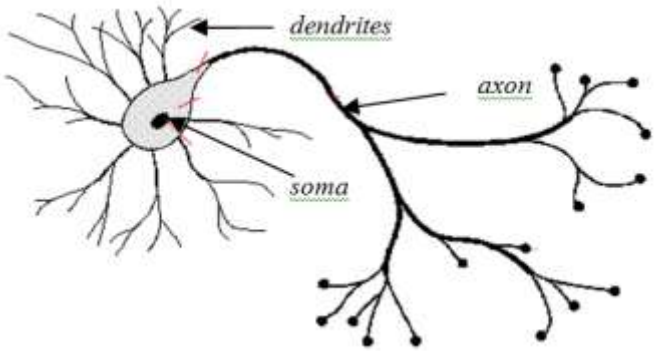


**Fig. 3:** A Neuron.

It must however be noted that all neurons in a cell interact with one another. This interaction happens only when there is to be an exchange of spikes. In doing this though, an axon may need to stretch a distance to reach the next neuron. The junction between two neurons is called ***synapse*** (Rodolfo, L., 2008).

***SN P System's Rules:***

Ordinarily, a spike is either moved, created or destroyed, within or outside a neuron, but could never be modified (because there is only one type of objects in the system, which is the spike). In a typical SN P environment, there are two ways of rules' application. These are: the *maximally parallel* and the *sequential* way. When two or more (sets of) rules can be applied in a given computation step, a nondeterministic choice is taken.

Therefore, this object called spike, evolves according to a set of *spiking rules* and *forgetting rules* each of which is associated with a neuron that uses the rules for sending or internally consuming the spikes.

***The Spiking Rule:***

The rules for spiking should take into account all spikes present in a neuron not only part of them, although not all spikes are consumed in this way. In other words, firing Rules allow a neuron to send information to another neuron in the form of electrical impulses/signals/spikes which are accumulated at the target neuron, consuming some of their own spikes.

So, going by the architecture of SN P system, objects are evolved by means of *spiking rules*, which are of the form: $E/a^c \rightarrow a; d$, where $E$ is a regular expression over {a} and $c, d$ are natural numbers, $c \geq 1$, $d \geq 0$. The meaning is that a neuron containing $k$ spikes such that $a^k \in L(E)$, $k \geq c$, can consume $c$ spikes and produce one spike, after a delay of $d$ steps. The produced spike is sent (maybe with a delay of some steps) to all neurons to which a synapse exists outgoing from the neuron where the rule was applied. However, a neuron called *output neuron* sends its own spike to the environment. Hence, this leads to the generation of *spike train* which is formed from the binary numbers 1s and 0s of the released spikes or otherwise.

***The Forgetting Rule:***

This removes all the spikes from the neurons and is of the form $a^s \rightarrow \lambda$ with the meaning that $s \geq 1$ spikes are removed, provided that the neuron contains exactly $s$ spikes. We say that the rules "cover" the neuron, all spikes are taken into consideration when using a rule. Forgetting rules imply that the system only consume the spikes, while no one is sent to the neighbouring neurons

In general therefore, an SN P system of degree m ≥ 1 is a construct of the form:

$\prod = (O, \sigma_1 \ldots . \sigma_m$ syn, out),

Where:

1. $O = \{a\}$ is the singleton alphabet. (a is called spike);
2. $\sigma_1, \ldots . , \sigma_m$ are neurons, of the form $\sigma_1 = (n_i, R_i)$, $1 \leq i \leq m$, where:

* $n_i \geq 0$ is the initial number of spikes contained by the neuron;
* $R_i$ is a finite set of rules of the following two forms:

(1) $E/a^c \rightarrow a$ ;d, where E is a regular expression over $O$, $c \geq 1$, and $d \geq 0$;

(2) $a^s \rightarrow \lambda$, for some s ≥1, with the restriction that as L€ for no rule $E/a^c$ a; d of type (1) from $R_i$;

3. syn $\subseteq \{1, 2, \ldots ., m\} \times \{1, 2, \ldots ., m\}$ with $(i, i) \in$ syn, for $1 \leq i \leq m$ (synapses);
4. out $\in \{1, 2, \ldots ., m\}$ indicates the output neuron

The rules of type (1) are called *spiking rules*, which is written in a shorthand notation as $a^c \rightarrow a^b$ . The rules of type (2) are called *forgetting rules*. The application of the rules depends on the contents of the neuron. This implies that the applicability of a rule is established based on the total number of spikes contained in the neuron.

If no firing rule can be applied in a neuron, there may be the possibility to apply a forgetting rule, which removes from the neuron a predefined number of spikes.

***Computation in SN P System:***

A *computation* starts from an initial configuration of the system and terminates when no other rule can be executed. This stage is called *halting configuration* and the result is obtained in the output neuron. A configuration of the system at a given time is the multiplicity of objects per region. Two major qualities of SN P which are being constantly explored are its computational completeness and computational efficiency.

The result of a SN P system is interpreted in either of the following two ways:

(i) To check the difference between when the time the first spike is produced at some time t and that of second spike at some other time t + k. So, k = (t+k)-t is obtained. (Used with generative SNP system i.e a system which sends spike to the environment)

(ii) Taking the result of the time difference of the output neuron as '1' when a spike is released and '0' when there none is produced. Invariably a binary spike train of 1s and 0s would be generated.

Consider the example given in [18] to generate the set of all odd integers $\{2k + 1|k \in N\}$

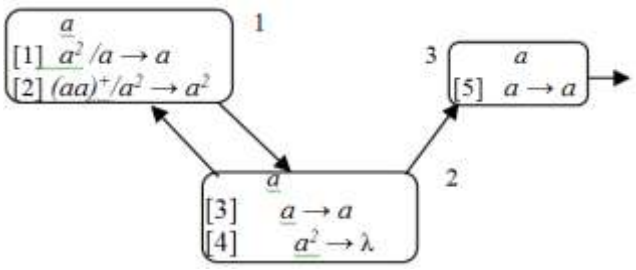


**Fig. 4:** SN P system computing $\{2k + 1|k \in N\}$.

In the figure 6 above, there are (i) three neurons 1, 2 and 3 as the output neuron (ii) five rules [1], [2], [3], [4] and [5] (iii) just one spike *a*. The time difference between the first spike produced by the output neuron and its succedding spikesgives the result of the computation.

The table X below depicts how the computation is carried out

**Table 1:** Computation steps in SN P system

| $t_0$ | t | t+1 | t+2 | t+3 | t+4 |
|---|---|---|---|---|---|
| | a | $a^2$ | a | $a^2$ | $\lambda$ |
| $\alpha_1 = 1$ | - | $r_1/r_2$ | - | $r_2/r_1$ | - |
| | a | a | a | $\lambda$ | $\lambda$ |
| | a | $\lambda$ | a | $\lambda$ | $a^2$ |
| $\alpha_2 = 1$ | $r_3$ | - | $r_3$ | - | $r_4$ |
| | $\lambda$ | $\lambda$ | $\lambda$ | - | $\lambda$ |
| | a | a | $\lambda$ | a | $\lambda$ |
| $\alpha_3 = 1$ | $r_5$ | $r_5$ | - | $r_5$ | - |
| | ! $\lambda$ | ! $\lambda$ | $\lambda$ | ! $\lambda$ | $\lambda$ |

At the initial time $t_0$, neuron $\alpha_1$, $\alpha_2$ and $\alpha_3$ were having a spike each. Consequent on the initial rule application, $\alpha_2$ fires a rule and spikes at time t. But because $\alpha_2$ has just one spike *a*, it could only employ rule $r_3$ ($a \rightarrow a$) which is interpreted as; consumes one spike, and produces a spike each to neurons $\alpha_1$ and $\alpha_3$) . However, despite the fact that at time t, $\alpha_1$ has a spike, no rule would be applied because a minimum of two spikes is required. At the same time t, $\alpha_3$ fires its rule r5 in which a spike is consumed and one spike is sent outwardly.

At the next stage, t+1, non-deterministically, $\alpha_1$ which now has 2 spikes, may choose between $r_1$ and $r_2$. If rule $r_1$ is applied (though $r_2$ could as well be applied), then a spike is consumed and 1 spike is also sent to neuron $\alpha_2$. Because $\alpha_3$ had earlier on gained a spike from $\alpha_2$. It employs $r_5$ again by sending a spike to the environment.

At the preceding steps, $\alpha_2$ gained a spike from $\alpha_1$ because of the application of rule $r_1$ which now forces $\alpha_1$ to be left with a spike. Please note, if $r_2$ had been engaged by neuron $\alpha_1$, then all its spikes would have been consumed thereby forcing $\alpha_2$ to have two spikes in the present t + 2 step. Implying that it would have compelled $\alpha_2$ to employ its forgetting rule $r_4$ (i.e $a^2 \rightarrow \lambda$). Hence, it forgets 2 spikes ($a^2$).

The whole computation would have halted in step t + 2 if $\alpha_1$ had used its $r_2$ because there wouldn't have been any rule left to employ. However, computation continues since $r_1$ was engaged. Since $\alpha_3$ had earlier on spiked at step t, the system computes its only result: (t +1) – t (which is odd number 1).

Conclusively, if $\alpha_1$ decides to use rule r1 non-deterministically as against rule $r_2$ then $\alpha_3$ spikes at every step t + 2k + 1, implying that the system computes the value (t + 2k + 1)- t = 2k + 1. Invariably, the system halts at the next step, once $\alpha_1$ uses $r_2$ instead of $r_1$.

***SN P System's Model and Simulation:***

***SN P System's Model:***

Figure 5 below depicts a simple theoretical model applicable for solving computationally hard problems in a typical SN P system. A problem submitted to the gate of an SN P system, is reducible to a framework where objects and other constituents are well defined, thereby leading to the application of appropriate rules. This is because a well-situated problem leads to better computation and subsequently yields reliable and efficient solution which is judiciously made use of by man.

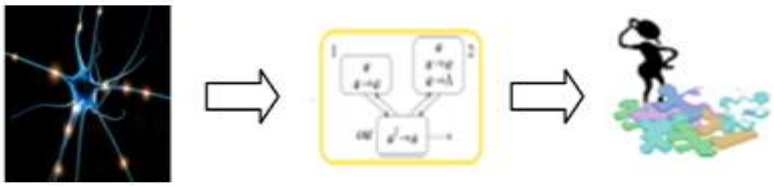

**Fig. 5:** SN P Theoretical Model for Problem –Solving.

***Simulating in SN P System:***

All the variants (cell-like, tissue-like and neural-like) of P-system share the same simulation structure. So, a typical p-system structure is made up of three parts., namely: (i) the input which comprises of the user and the definition of the P-system to be executed (ii) The core – which is the computer system's processor where the manipulation of the P system earlier defined takes place (iii) output- where relevant information is generated and given out to the user (See Figure 6).

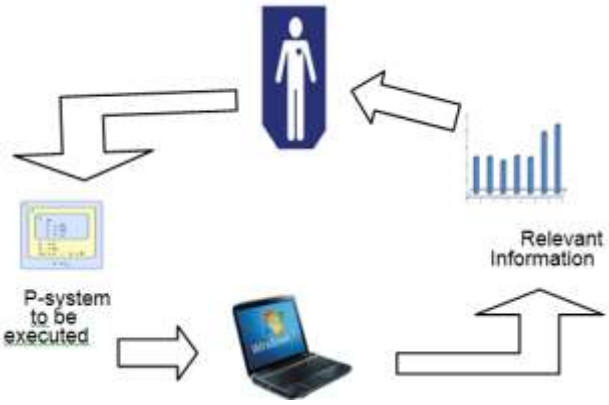


**Fig. 6:** Structure of a P-System Simulator.

The simulation algorithm for SN P System is presented thus:

I. Initialization: Initialize Data Structures

II. Rule Selection: Compute the Rule Set to be executed in the current step

III. Build execution sets: Split the Rule Set according to the kind of its components

IV. Execute division and budding rules

- Compute new neurons out of existing ones by applying division and budding rules.
- Add new synapses according to the synapse dictionary.

V. Execute spiking rules

VI. Ending

- Update the current configuration with the newly created one
- Check the halting condition (no more rules are applicable).

***Positive Impacts of SN P System On IDS:***

Spiking Neural P systems are a versatile formal model of computation that can be used for designing efficient parallel algorithms for solving known computer science problems. This is usually achieved by introducing features to exponentially increase the working space in a polynomial time. From the viewpoint of real-world applications, Jun Wang & Hong Peng (2010) opined that SN P system is highly attractive based on the following reasons:

- parallel computing advantage,
- high understandability (due to their directed graph structure),
- dynamic feature (neurons firing and spiking mechanisms make them suitable to model dynamic behaviors of a system),
- synchronization (that makes them suitable to describe concurrent events or activities),
- non-linearly (which makes SN P systems suitable to process non-linear situation). This is because if a bound is imposed on the number of spikes present in any neuron during a computation, then a characterization of semilinear set of numbers is obtained.

More importantly however, within any parallel environment, neuron supports three kinds of parallel processing. These include:

a.) Multiple simulations distributed over multiple processors, each processor executing its own simulation.
b.) Distributed network models with gap junctions.
c.) Distributed models of individual cells (each processor handles part of the cell). Setting up distributed models of individual cells may somehow require considerable effort.

Specifically therefore, the parallel computing benefit of SN P system's paradigm could be applied to improve the intelligent algorithms (such as Bee Algorithm) used for feature subset selection in IDS. Hence, when this is done, a more efficient system which would be able to detect attacks in real-time would be obtained.

***Conclusion:***

So far, this presentation had dwelt on the preview of intrusion Detection System in general and Spiking Neural P system in particular. Their converging point was established. Also, SN P rules and formalism were explained. More importantly however, effort was made in analyzing a neuron and how neuronal potentials (spike trains) are generated, transferred. The computational capabilities of neurons were also not left out. Finally, area through which SN P system could be used in IDS modeling was pin-pointed.

In order to effectively address the various intrusion threats, a system should combine several intrusion detection approaches.

## REFERENCES

Bace, R.G., 2000. *Intrusion Detection: Defining Intrusion Detection.* Macmillan Technical Publishing.

Chen, H., T.O. Ishdorj, G. Păun and M.J. Perez-Jimenez, 2006. "Spiking Neural P systems with extended rules". In *Proc. Fourth Brainstorming Week on Membrane Computing*, Sevilla, RGNC Report 02, 241-265.

Debar, H., M. Dacier and A. Wespi, 1999. "*Towards a Taxonomy of Intrusion-Detection Systems"* Computer Networks, 31(8): 805-822.

Francis George Carreon-Cabarle Spiking Neural P Systems: Implementations and Applications. Unpublished master of Science thesis, University of the Philippines April 2012.

Freund, R. and M. Oswald, 2007. "*Spiking Neural P Systems with inhibitory axons"*. *AROBConf.*, Japan.

Garcia-Teodoro, P., J. Diaz-Verdejo, G. Macia-Fernandez and E. Vazquez, 2009. "*Anomaly-Based Network Intrusion Detection: Techniques, systems and Challenges".* Computers & Security, 28: 18-28.

Gerstner, W. and W.M. Kistler, 2002. *Spiking Neuron Models–Single Neurons, Populations, Plasticity* Cambridge University Press.

Gh. Păun, M.J. Pérez-Jiméenez, G. Rozenberg, 2007. Computing morphisms by spiking neural P systems. *Intern. J. Found. Computer Sci.*, 18(6): 1371-1382.

Hoffmeyer, J., 1998. "*Surfaces Inside Surfaces. On the Origin of Agency and Life"*. Cybernetics and Human Knowing, 5(1): 33-42.

Ionescu, M., 2008. "*Some Applications of Spiking Neural P Systems*", Computing and Informatics, 27: 3.

Ionescu, M., G. Păun and T. Yokomori, 2006. "Spiking Neural P systems". Fundamenta Informaticae, 71(2-3): 279-308.

Luis F. Macías–Ramos,Ignacio Pérez–Hurtado, Manuel García–Quismondo, Luis Valencia–Cabrera, Mario J. Pérez–Jiménez Agustín Riscos–Núñez A P-Lingua based simulator for SN P Systems. Research group on Natural Computing, University of Seville (2011).

Northcutt, S. and J. Novak, 2002. *Network Intrusion Detection* 3rd ed. New Readers Publishers, U.S.A.

Păun, G., 2002. *Membrane Computing- An Introduction*, Springer-Verlag, Heidelberg.

Rodolfo, L., 2008. *Neuron*. Scholarpedia, 3(8): 1490.

Some applications of Spiking Neural P Systems Computing and Informatics, 27: 515-528. 2008.

Tanenbaum, A., 2000. *Computer Networks*, 3rd Edition, Prentice Hall.

Teresa, F.L., 1993. "*A Survey of Intrusion Detection. Techniques"* Computer & Security, 12: 405-418.

The P Systems Website: http://ppage.psystems.eu.

Torrano, C., A. Perez-Villegas, G. Alvarez, 2010. An Anomaly-Based Approach for Intrusion Detection in Web Traffic. *Direct*, 5: 446-454.

Wang, J. and H. Peng, 2010. "Adaptive Fuzzy Spiking Neural P systems For Fuzzy Inference and Learning". In *Proc. Eight Brainstorming Week on Membrane Computing*, Sevilla, Spain, pp: 235-248.