Autonomic Computing applied to Network Security: A Survey

Ariel S. Teles, Jean P. M. Mendes, Zair Abdelouahab
Department of Electrical Engineering, Federal University of Maranhão
São Luís – MA – Brazil
{arielsoaresteles, jeancomp}@gmail.com, zair@dee.ufma.br

Abstract— **The constant increase in the number of computer network attack attempts has pushed researchers community to devise better security strategies. However, the rapid growth both in quantity and complexity of components and services offered in today's networks has increased the difficulty of administering these, making approaches based only on human interventions impracticable. In order to circumvent this problem, a modern approach called Autonomic Computing (AC) has gained attention from researchers related to network security management. AC has the essence of self-management and the implementation of its concepts for network security systems introduces the ability of self-assurance. This paper aims to introduce the concepts of AC and shows their applicability to the context of security in computer networks.**

Index Terms— **Autonomic Computing, Intrusion Detection, Network Security.**

I. Introduction

SECURITY in computer networks is an area that consists of protecting data during transit against its unexpected changes, unauthorized access and unavailability. Since the advent of Internet, several research works for better security strategies have increased considerably due to a large number of attempted attacks that have been carried out. These attacks, when successful, have caused financial and image loss to companies, institutions and individuals.

There are several obstacles to be faced to achieve truly secure networks; among them we can highlight the existence of dependence of security systems management with human intervention, which is a continuous process that increases the level of difficulty. Another example of obstacles is that attacks on computer systems are becoming increasingly sophisticated and there are several deficiencies in current security systems.

Thus, the problem of security management is becoming more complex and it is therefore interesting to use resources offered by the Autonomic Computing (AC). AC systems are able to manage themselves and dynamically adapt to changes in order to restore the balance in agreement with the policies and business goals. To do this, AC has effective mechanisms that allow them to monitor, control and regulate themselves, and recover from problems without recourse to external intervention.

The architecture and properties of AC provide systems with advantages to network security. Besides showing the intrinsic characteristics of self-management, an autonomic element provides other features that can be used to solve particular problems of network security such as learning techniques and cooperation between applications.

This paper discusses the concepts of AC and shows their applicability to the context of computer network security. Applying the concepts of AC in network security introduces the ability of self-security, through services and security management functions that are performed without the need for a human manager, by just defining the objectives and initial parameters provided by the administrators.

The rest of this paper is organized as follows. All the fundamentals of AC are described in Section 2, showing the properties and the architecture. Section 3 describes some concepts of network security and shows the problems which may be faced. The need for autonomic mechanisms in computer network security and related works are detailed in sections 4 and 5 respectively. Finally, section 6 gives the conclusions of this work.

II. Autonomic Computing

The term Autonomic Computing was founded in 2001 with a manifesto published by Paul Horn, IBM researcher, who issued a challenge on the problem of managing the growing complexity of software [1]. The term autonomic comes from biology and it is related to involuntary physiological reactions of the nervous system [2]. In the human body, the autonomic nervous system takes care of unconscious reflexes, i.e. body functions that do not require our attention as the expansion and contraction of the pupil, digestive functions of the stomach and intestine, the frequency and depth of breathing, dilatation and constriction of blood vessels, etc. This system reacts to changes or disturbances caused by the environment through a series of modifications in order to contain the disruption to its internal balance.

In analogy to human behavior, it is said that a computer system is in equilibrated state when its internal environment (formed by its subsystems and the system itself) is in due

The development of this work is made possible by financial support provided by the Foundation for Scientific Research and Technological Development of Maranhão (FAPEMA) – Brazil.

proportion with the external environment. Parashar and Hariri [3], and with more details in [4], have noted that if the internal or external environment disturbs the stability of the system, it shall always act in order to restore the original balance. Thus, AC systems are systems that can manage themselves and adapt dynamically to changes in order to restore balance in accordance with the policies and business objectives of the system [5]. They must have effective mechanisms to enable them to monitor, control and regulate themselves, and recover from problems without the need for external intervention.

A. Properties of Autonomic System

The essence of AC is self-management. To implement it, the system must be aware of itself and its environment. Thus, the system must know accurately its own situation and its operational environment in which it operates. From a practical standpoint, of Hariri [5], the term autonomic computing has been used to denote systems that have the following properties:

- **Self-awareness**: The system knows itself: its components and their interrelationships, their state and behavior;
- **Context-aware**: The system must be aware of the context of its execution environment and be able to react to changes in its environment;
- **Self-configuring**: The system should dynamically adjusts its resources based on their status and the state of the execution environment;
- **Self-optimizing**: The system is able to detect performance degradations and perform functions for self-optimization;
- **Self-protecting**: The system is able to detect and protect resources from internal and external attackers, keeping their safety and overall health;
- **Self-healing**: The system must have the ability to identify potential problems and reconfigure itself in order to continue operating normally;
- **Open**: The system should be portable to different hardware architectures and software and therefore must be built on open protocols, interfaces and standards;
- **Anticipatory**: The system must be able to anticipate as far as possible, considering the needs and behaviors of its context and manage them in a pro-active way.

B. The Architecture of an Autonomic System

Autonomic system architectures aim at formalizing a framework that identifies the common functions and lays the foundation necessary to achieve autonomy. In general, these architectures provide solutions to automate systems management cycle, which involve the following activities:

- **Monitoring or Measuring**: Collects, aggregates, correlates and filters data of managed resources;
- **Planning and Analysis**: Analyzes the data collected and determines if changes should be made in the strategies used by the managed resource;
- **Control and Enforcement**: schedules and executes the changes identified as necessary for the function of analysis and decision.

C. MAPE-K

In 2003, IBM proposed an automated version of the cycle system management called MAPE-K (Monitor, Analyze, Plan, Execute, Knowledge) [1], represented in Figure 1. This model is increasingly used to inter-relate the architectural components of autonomic systems. An autonomic system consists of a set of autonomic elements.

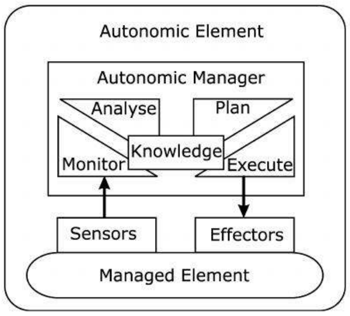

Fig. 1. MAPE-K: Autonomic Management Cycle [7].

An autonomic element contains a single autonomic manager that represents and monitors one or more managed elements (hardware component or software) [6]. Each autonomic element acts as a manager responsible for promoting resource productivity and quality of services provided by the system component in which it is installed. In the MAPE-K autonomic loop, the managed element represents any software or hardware feature which is given by the autonomic behavior of an autonomic manager coupling.

The sensors are responsible for collecting information from the managed element. These data may be diverse, for example, the response time of requests from customers, if the managed element is a Web server. The information collected by the sensors are sent to the Monitor where they are interpreted, preprocessed and placed in a level of abstraction, and then sent to the next step in the cycle, i.e. Analyze and Plan. In this stage, there is a kind of product which is a work plan, which consists of a set of actions to be executed by the Execute. The component responsible for making the changes in the environment is called Effectors.

Only sensors and effectors have direct access to the managed element. Throughout autonomic management cycle, there may be a need for decision making, thus it is necessary for the presence of a knowledge base (knowledge), and this is the most commonly exploited in the process of analysis “Analyze” and planning “Plan”.

This is implemented using two or more autonomic management cycles, one or more local cycles and one global control cycle. The cycles of local control deal only with known states of the local environment, based on the knowledge found in their own managed element. For this reason, the local cycle is unable to control the overall behavior of the system. The global cycle, in turn, based on data from the local managers or through a global monitoring, can make decisions and act globally on the system. However, the implementation of interactions among various levels depends on existing needs and limitations of the application.

III. Networks Security

The large growth in the number of components and services in modern computer networks has increased the level of complexity for their management and administration. Several devices are integrated to networks that require connectivity anytime and anywhere, and are characterized by their heterogeneity.

Thus, it is necessary to apply the idea of AC to Computer Networks, by giving the ability of self management, known as Autonomic Networks. The services and network management functions are performed transparently to its users without necessity for a human manager, taking into account only the goals and initial parameters of the system. The network must be able to learn from the actions of its components by analyzing the results. The adaptability and learning are characteristics of autonomic networks.

In this scenario, one cannot leave the information security issue, which is prerequisite for a proper function of any computer system, by taking all measures aimed at preserving and protecting information. Since these networks are almost always connected to the Internet, applications running on them may suffer malicious activity originating from any connected user.

Accesses to the World Wide Web offer the possibility of discovering and exploiting vulnerabilities very quickly, almost always quicker than the upgrade of security tools and patches issued by software manufacturers. Thus, a number of security incidents are growing very rapidly causing an amount of damages. However, there is also a growing research for devising new mechanisms and techniques to increase the security level. Access policies, use of firewalls, intrusion detection systems, honeypots, among others, are some of these measures.

In information security, it is possible to identify the following basic properties [8]: confidentiality, integrity, availability, authenticity, non-repudiation, auditing and regulation. These properties guide for the determination of a focus in which a particular application should be developed to meet the security requirements specified by the necessity of the proposed environment. It is also important to define the phases of protection where it will operate.

A. *Steps to Protect Networks*

It is possible to highlight four phases in order to protect the network against attacks [9]: Prevention, Detection, Forensics and Defense, as seen in Figure 2.

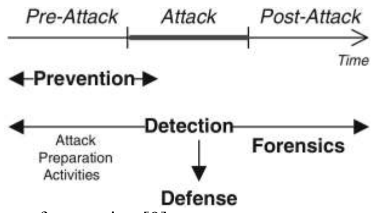

Fig. 2. Four phases of protection [9].

Prevention comprises all methods used to prevent attacks in order to ensure confidentiality and data integrity with the use of access controls to network resources. It includes techniques for authorization and authentication (login services), building trust, as well as encryption and traffic filtering (using firewall). It is important to emphasize that prevention is only possible for known attacks, but there is work being developed to predict the occurrence of unknown attacks, [10] and [11].

Prevention mechanisms are considered defense systems, in the first line, i.e., they are responsible for the first step in securing a computer network. Normally this defense in the first line is made in the design phase of the network in order to develop it as safely and get better results when put into operation. These mechanisms are typically implemented to control access to resources and information in the network.

If prevention fails, detection is the next phase to deal with an incident. Detection is then the discovery process of an attack or preparation of an attack, or any other malicious activities, from a port scan evidence to crack passwords by brute-force technique. This is usually done by analyzing data captured by a sniffer, through interception and recording of data traffic over the network.

Intrusion Detection Systems (IDSs) are used in the detection phase. When they perform network monitoring and thus are called Network Based Intrusion Detection System (NIDS), or when connected to a device, are known as Host Based Intrusion Detection System (HIDS). Their purpose is to find the occurrence of an attack or malicious activity. An IDS may use several approaches to identify an attack, the best known are: Signature Based and Anomaly Based [12].

Another mechanism used in the detection phase is known by the name of Methods of Deception, as defined in [13] by creating a fake environment to fool malicious users. Techniques are used in which the attacker interacts with a feature set as a trap, intentionally vulnerable, which emulates services or systems that really should be the target of its action. The technique that is widely used is with the use of honeypots, defined by Spitzner [14] as a network resource whose function is to be probed, attacked or compromised. This means that a machine can be invaded, and this one is configured to obtain information about the attacker. The intention is that the intruder when performing an attempted invasion, in which the network has a honeypot running, has the feeling that he is interacting with a machine that has some functionality and he can get some use.

If a malicious traffic is detected, then the process of defense is initiated. To achieve this goal, it is necessary that the system implements these two phases (detection and defense), integrating various security systems. An example is the Intrusion Prevention System (IPS), which generates some response in order to neutralize the attack and may include an IDS and a firewall.

In some cases when the earlier phases of prevention and detection fail and the attack was successful, it is necessary to make an analysis of all the logs in order to learn how the methods used in the processes of detection and defense can be improved to prevent future incidents. This phase is called forensics, which aims to conduct an investigation to find out specific details of attacks, with results that in some way contribute to the improvement of network protection. For ES Pilli et al. [15], techniques for network forensics provide resources for researchers track down the attackers.

From the outline of these phases it is possible to characterize the functions of network security systems. However, many of these systems have currently some limitations and problems. Along with this, attacks on computer networks are becoming increasingly sophisticated. These issues are detailed below.

B. *Problems Faced*

Zseby et al. in [9] state that the security of networks crucially requires greater attention by the administrator and almost always more effort and cost, as new protocols and applications introduce new vulnerabilities. Unfortunately, the mechanisms used to increase the level of security currently have some problems, namely: solid, low defense skill, without self-adaptation, without self-evolution and without self-learning.

To Atay and Masera, in [16], all methods of analysis of threats, vulnerabilities and risks need to continually update their knowledge of the weaknesses found in the new network assets. This serves to identify how these weaknesses can be exploited to further define and implement the necessary countermeasures. This is a continuous cycle, as new evaluations are needed over time.

However, it is known that information about new attacks is not immediately disclosed by manufacturers or by the community that develops the software, due to its sensitivity. This is because this information can be used to further explore the vulnerabilities, since malicious users can obtain them. They are soon published after the manufacturers release patches. The methods of risk analysis cited by [16] should redo the safety assessment of systems taking into account information about new attacks when they are released, or the use of intelligence techniques to detect them even before the disclosure.

In this situation, Wang et al. [17] states that the right direction for the development of applications in defense and security is the adoption of two features: the integration and intelligence. Integration is to enable management of multiple protective features in a distributed network environment, and the intelligence adapted to the environment is to increase the efficiency of protection based on its knowledge that it has acquired, and finally, achieving a balance between the security application and network environment.

Allied to the problems faced by today's security systems, attacks on computer systems are becoming increasingly sophisticated, unpredictable, and often with a greater number of sources [16]. Examples of these attacks can be carried out to highlight the use of botnets [18] and 0-day attacks [10] [11].

IV. Networks Autonomic Security

The protection of today's networks systems are based on the paradigm of interactive computing, i.e., it is left to human administrators decide what to do and how to protect systems in the event of malicious attacks or cascading unexpected errors. Systems that incorporate more than one phase of network protection, aiming to further increase the security level, are more complex. This is due to the fact that they have more components to meet the requirements of each phase. This complexity requires a constant human intervention specialist for the correct use of the system.

Thus, it is interesting the use of autonomic mechanisms to automate the management processes. Applying the concepts of AC network security to a system will provide the latter with the ability of self-security, through which services and security management functions are performed without the need for a human manager, considering only the objectives and initial parameters set by their administrators.

It is also possible to highlight that in order to try to propose solutions to security problems faced by computer networks, autonomic system architecture can be applied to software development focused on defense and security. The MAPE-K model provides a conceptual view of how autonomic systems can be developed to meet security needs.

Sensors can be any program that checks for occurrences of malicious traffic, regardless of what stage of protection is, collecting relevant information from the network to be sent to the monitors. Example of data collected can be, for example, traffic to honeypots, IDS alerts, firewall logs, etc. The monitors receives such information and treats them to extract what is relevant, for example, the source IP address of the intruder, the protocol used by the department in which the attack was carried out, time / date of the intrusion, etc.

The monitors send the necessary information to the analysis and planning components where these will use it for processing. However, the phases of analysis and planning may be implemented in a single component. The processing performed by this component varies according to the strategy adopted by the autonomic manager who set the objectives for the system. An example of processing using ECA rules [19] can be seen in Figure 3. ECA (event-condition-action) rules are declarative specifications of regulations that govern the behavior of application components. For each event, it is a defined set of rules that can generate one or more actions. In this example, in case the IDS gives an alert (*IDSAlert*) and if the source IP that generated the alert is not blacklisted in the firewall´s then a script that contains *SrcIpAddr* adds the IP address (i.e. *Add SrcIpAddr in BlackList*) to the blacklist.

on *IDSAlert* if *BlackList !Contains SrcIpAddr* do *AddSrcIpAddr in BlackList*

Fig. 3. Example of reconfiguration.

The knowledge base can be used by the component analysis and planning with strategies to prevent performing actions previously realized. For example, Figure 3 shows that IP addresses already entered in the backlist will not be included again inside that list. The component analysis and planning produces a plan of actions to be executed by the component execute, which consist of adding IP source address that generated the alert in the blacklist, as shown in figure 3.

The component execute applies the actions on the managed element through the effectors. The effectors are responsible for making configuration changes to the managed element, or in any application or in a network security. The goal in making configuration changes is to increase the level of security. In Figure 3, the effectors interacts directly with the application responsible for the blacklist, i.e. the firewall.

In addition to the architecture offered by the AC, as well as the state that current security systems should achieve and also the growing of attacks on computer networks, it is possible to specify the type of properties of autonomic systems that are needed [20], as seen below:

- **Self-protecting**: Refers to the ownership of the system to defend itself from accidental or malicious attacks. Thus, the system must have knowledge of potential threats, and provides mechanisms to address them. To achieve this property the system must have the ability to anticipate, detect, identify and protect against threats;
- **Self-healing**: Responsible for identifying and correcting errors or failures. In the context of network security, the autonomic system should be able to detect, diagnose and repair problems resulting from attacks on production assets of the network. Using the knowledge about the configuration of network resources, the system must have a component that must analyze diagnostic information showing the occurrence of faults or damage caused by a network attack, and later seeks a solution to be taken, and apply it and then test whether it was satisfactory. Importantly, the healing process should be conducted with maximum transparency for legitimate users of the network;
- **Self-configuring**: This property provides systems with the ability to automatically configure and reconfigure according to business policies provided by their managers, which define what must be done and not how. In order to automate the configuration management, a security system must have dynamic reconfiguration capability with minimal human intervention;
- **Self-learning**: A Property that provides systems with the ability to learn from and sense data from experiments and results obtained in previous actions. It is a fundamental property for security systems, since it provides the ability for the system to learn to defend against previously unknown, or at least recognize malicious traffic in the network for further defense.

Although AC offers various qualities, applying these to computer networks and their security is not a trivial process. There are challenges to be faced to achieve self-management. According to Agoulmine et al. in [21], the challenge is to simplify the management task for the administrator by automating the decision process. Security issues are another challenge for the development of autonomic systems, whereas the use of self-learning and self-evolution may cause loss of control under the management of human decisions made by the system itself, with a possibility of deviation from the initial goals that are set. Thus, in the process of developing autonomic software a validation phase should be applied rigorously.

V. AUTONOMIC SYSTEMS OF NETWORKS SECURITY

Below are shown some works that use the AC as a basis for the development of their proposals or implement any of the properties offered by AC.

A. ISDS

In [17] was developed Intelligent Security Defensive Software (ISDS), a model and security software based on AC. ISDS's strategy is to make the process of building software security by giving it intelligence. In other words, build a model using the ISDS software is to make its components change dynamically according to the current security situation of the network. For this, the ISDS provides awareness of the context.

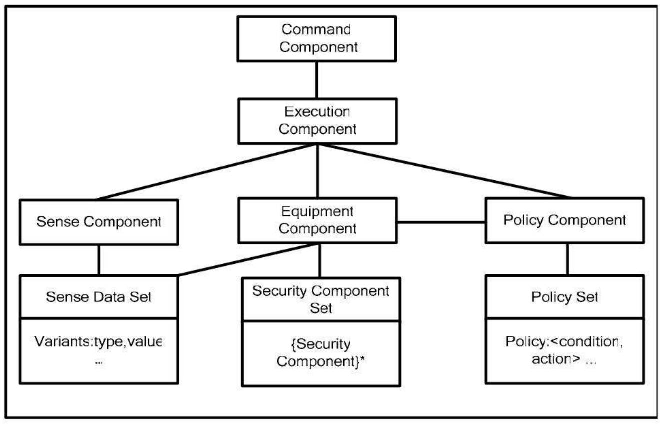

Fig. 4. The Conceptual Framework of an ISDS [17].

The ISDS model is a distributed defense system characterized by being flexible and self-adapting. The areas where it can be applied are mainly in the management of security on local networks and management of information security on the Internet. The idea of the model is that the system can analyze the information from the environment and adjust its structure dynamically. It consists of some basic components which are: command, execution, sensing, policy, and other auxiliary equipment as shown in Figure 4. The component command is the main part of the ISDS, with the functions of: activation of policy components, sense equipment and implementation of decision making based on the sensing component, the management of all components of security and policy update of new policies and security components, verifying and resolving policy conflicts. The execution component works as a channel of communication between security entities and the component command. It takes care of filter, process and transmits the message to all other components, making them work properly. The component policies are primarily responsible to maintain the set of policies, decision-making of appropriate policies and also the passage of the outcome of the decision. Information on the outcome of the decision made is then passed to the equipment component to perform some action. Finally, the component environmental sensor collects information and sends it to the component command, taking into account the cost of the two sensing modes: emergence and timing, where the first has the highest priority over the second.

B. Autonomic Security and Self-Protection based on Feature-Recognition with Virtual Neurons

In the work developed in [22], the authors have presented an autonomic security mechanism based on virtual neurons and the recognition features. Their approach works to automatically detect many security problems that are currently difficult to make defense. Through simulation and different case studies, results show that this solution is feasible. The virtual neurons are developed similarly to the neurons of animals, as seen in Figure 5. The idea is that they are distributed in a way to perceive changes in the environment and react to stimulus. A virtual neuron is actually a simple software capable of context-aware in order to analyze the context information and perceive possible appearances of attacks.

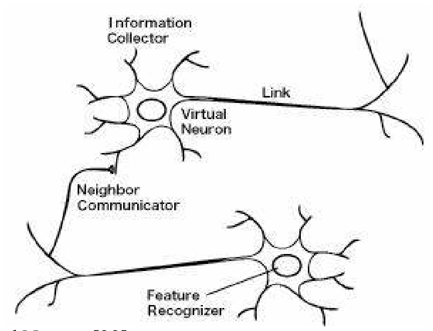

Fig. 5. Virtual Neuron [22].

In the virtual neuron, the Information Collector component captures various contexts information, such as memory and processor usage, process status information and network traffic. Neighbor neurons are those that connect directly through the Neighbor Communicator. There is another component called the Feature Recognizer, and its operation is based on knowledge of information that characterizes an attack (based on subscriptions). This information is passed to the Information Collector Feature Recognizer by the neuron itself and to its neighbors through the Neighbor Communicator. Virtual neurons can be easily distributed, because the installation package is compact, they require few computational resources and are easy to install on the hosts.

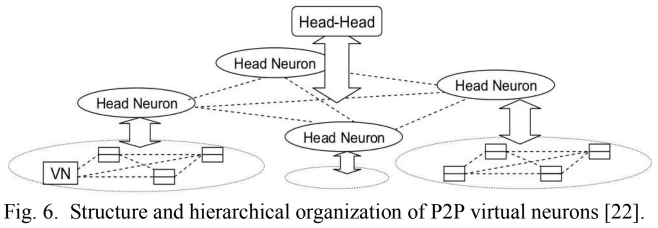

Fig. 6. Structure and hierarchical organization of P2P virtual neurons [22].

These neurons are distributed in a virtual hierarchical architecture and Peer-to-peer (P2P), as seen in Figure 6. The structure-based P2P operates in relations with neighboring neurons, and their communication via message passing. The hierarchical architecture is used to increase efficiency in the propagation of messages by group of neurons, called cells. In each cell a neuron-neuron is elected as leader, the election algorithm takes into account the computational resources and communication speed of the host.

In the hierarchical organization, a neuron-chief is elected as the head of high-level leaders to other neurons. This procedure is repeated until there is a single neuron Chief higher on others. If a fault occurs somewhere in chief of a neuron cell, another neuron of the same cell will assume the role of neuron-chief, through a new election. In this hierarchical organization of messages are delivered more efficiently. The security mechanism employed by this distributed system is to detect illegal messages or data.

Origins of attacks are traced to identify and locate the attacker. After identifying the attacker, all neurons receive messages to discard traffic originating from that malicious user. The reconfiguration of resources is made at this time to achieve the system defense. To perform the detection and subsequent defense, a mechanism is developed (characteristic or signature) for each type of attack, such as: Eavesdropping, Replay, Masquerading, spoofing and Denial of Service (DoS).

C. Self-Configuration of Network Security

The work [23] presents an approach of self-configuring to control and manage the security mechanisms in large networks. It automatically configures the system security mechanisms and modifies the resource settings and policies at runtime. To show its feasibility, the authors have implemented an Autonomic Defense System Network (AND). AND is a system that can detect network attacks, known or unknown (me be last-day-attacks) and proactively prevent or minimize impacts on network services.

The AND was developed on the framework AUTONOMIA [5], which is an extension focusing on strategies and mechanisms to detect and protect themselves from network attacks. The units have two software modules, they are: Component Management Interface (CMI) and Component Runtime Manager (CRM). The CMI is used to specify the settings and policies associated with each component (hardware or software). The CRM manages the operations of components using the policies set by CMI. More details about the framework AUTONOMIA may be seen in [24]. The main components of the AND are seen in Figure 7.

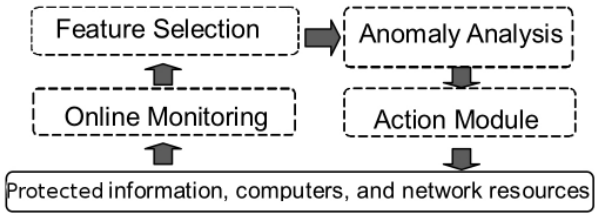

Fig. 7. The Architecture of AND [23].

We can observe that this architecture is based on the MAPE-K. The online monitoring collects data network traffic and transactions on computers through specific tools and log files generated by operating systems. The model selection feature filters the monitored data in order to find relevant information or characteristics to be passed to the next module. The anomaly analysis module uses a function to quantify whether there is a deviation from the standard profile of a system or network. If any is considered an anomaly then the action module executes the appropriate actions. The action module briefly restricts access to the attacked resources and then later tries to uninstall malicious code installed on compromised computers in the network.

D. *Quality of Protection*

Work started in [25] resulted in a new term in [26] called the Quality of Protection (QoP). It is a framework for proactive network defense that can be integrated with existing protocols for Quality of Service (QoS). The goal is to provide differentiated services for traffic flows according to their degree of abnormality. For this purpose, it was created a new metric called distance abnormality (DA), as seen in Figure 8, which can be used to classify traffic as normal, uncertain and abnormal. There is a Delta function that shows how much closer the traffic is normal or abnormal, and then it can be classified as probably normal or probably abnormal.

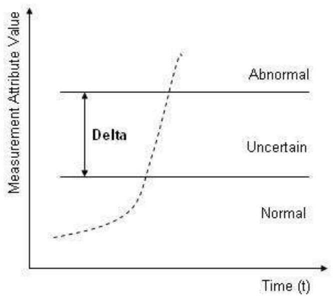

Fig. 8. The Distance of Abnormality [25].

The idea is that the DA metric is used in conjunction with QoS protocols to increase the priority of traffic considered normal and decrease of abnormal traffic. Tests were carried out on attacks of Distributed Denial of Service and worm propagation. This has been possible to demonstrate how the proposed approach can detect and reduce the impact caused.

E. *Properties in Autonomic Computing Systems Security*

Some works are characterized by providing some of the autonomic properties in isolation, although not based on the AC. That is, they were developed without following the concepts of AC, but provide some mechanism that qualifies in some of the autonomic properties. Systems like these are not regarded as autonomic.

It was developed in [27] a security operation to update firewall rules based on traffic to a honeynet [14]. In the scheme there is a module that analyses data to discover new attacks. This module analyzes traffic logs generated by the honeynet and uses data mining techniques to create dynamically new firewall rules passing them to the learning module which in turn filters out the inconsistencies between the rules and finally applies them. Thus the firewall continues increasing its strategies by improving its ability to defend the system against new attacks.

The tool Honeycomb [28] aims to automate the generation of attack signatures for intrusion detection systems from logs generated by honeypots. Actually this tool is a plugin for the honeyd [29], which is a framework of low-interaction honeypots [14]. Honeycomb generates signatures format Bro [30] and Snort [31]. Its intention is to create attack signatures at runtime, an activity that is usually done manually by security experts.

In the case of SweetBait [32], it was developed an automated system that uses low-and high interactivity honeypots to protect, recognize and capture malicious traffic. Based on the logs generated after discarding patterns in the white list, it automatically creates attack signatures, component implemented using Honeycomb. To provide a short response time to an attack it immediately distributes signatures to IDSs after its generation. Parallel to this, the signatures are continuously refined to increase accuracy and reduce false positives. This work is extended in Argos [10] that employs a broader feature, the property of self-protection by proactively react against attacks.

F. *Comparison*

The Table I illustrates a comparison between security systems for computer networks seen before. This comparison takes into account if the system is based on AC in its development, as well as the autonomic properties that the system offers as a resource.

TABLE I
AUTONOMIC NETWORKS SECURITY SYSTEMS

Properties / Type	Based in AC	Not based in AC
Self-configuring	[17] [22] [23] [26]	[27] [32] [10]
Self-optimizing	[17] [23] [26]	[28] [32] [10]
Self-healing	[23]	[10]
Self-protecting	[17] [22] [23]	[10]
Self-learning	[17] [23] [26]	[27] [28] [32] [10]

To achieve the original vision of the term Autonomic Computing the properties of self-configuring, self-optimization, self-healing and self-protection are sufficient [2]. To incorporate the properties of self-optimization and self-healing mechanisms of self-awareness, context awareness and self-configuring should be the system requirements. In particular in security systems, self-learning property is fundamental [33].

VI. CONCLUSION

In 2001, IBM produced a manifesto in which it warned of the difficulty of managing computer systems and pointed out the current range of systems as an alternative for solving this problem. This paper has presented a definition and main characteristics of a new approach to the development of systems that provide autonomy, Autonomic Computing. With the new approach, it involves a change in the way of designing computer systems.

The idea behind this approach is to develop self-managing software, with little or no human intervention. It is based only on high-level policies set by the supervisor and the knowledge acquired over time. A set of autonomic properties are used in this approach to reduce or eliminate human intervention in the management of computer systems, such as self-healing, self-protection, self-optimizing, self-learning, self-configuration, etc. In this view, the task of management is placed under the responsibility of the machines themselves.

We have shown in this paper that computer networks are scenarios where AC can be easily applied, mainly resulting from the growth of the Internet. In particular, we have presented the applicability of AC in a very specific environment, the management of network security. We have explained the needs of network security for autonomic mechanisms and how they can be implemented. Several related works done within the area were described to provide a more practical view.

Finally, research directions for network and application security are the adoption and integration of intelligence. The resources provided by AC are the most viable way to solve problems in computer networks and more specifically, in case of network security.

REFERENCES

[1] IBM, "An architectural blueprint for autonomic computing," *IBM Press*, in *IBM Press*, 3nd ed., Ed. New York: Prentice-Hall, 2005.

[2] R. Murch, "Autonomic computing," IBM Press, Prentice-Hall, 2004.

[3] M. Parashar, and S. Hariri, "Autonomic computing: An overview," *Unconventional Programming Paradigms*, Springer Verlag, 2005, pp. 247-259.

[4] M. Parashar, and S. Hariri, "Autonomic computing: concepts, infrastructure, and applications," CRC Press, 2007.

[5] S. Hariri, B. Khargharia, H. Chen, J. Yang, Y. Zhang, M. Parashar, and H. Liu, "The autonomic computing paradigm," *Cluster Computing*, Springer, vol. 9, 2006, pp. 5-17.

[6] J.O. Kephart, and D.M. Chess, "The Vision of autonomic computing," *Computer*, IEEE Computer Society, 2003, pp. 41-50.

[7] M. C. Huebscher, and J. A. McCann, "A survey of autonomic computing - degrees, models, and applications," *ACM Computing Surveys*, vol. 40, n. 3, 2008.

[8] M. Krause, and H. F. Tipton, "Handbook of Information Security Management," Auerbach Publications, 1999. Available: http://www.cccure.org/Documents/HISM/.

[9] T. Zseby, H. Pfeffer and S. Steglich, "Concepts for Self-Protection," *Autonomic Computing and Networking*, Springer Science, 2009, pp. 355-380.

[10] G. Portokalidis, A. Slowinska, and H. Bos, "Argos: an Emulator for Fingerprinting Zero-Day Attacks for advertised honeypots with automatic signature generation," in *Proceedings of the EuroSys*, 2006, pp. 15-27.

[11] J. Song, Y. Kwon, and H. Takakura, "A Generalized Feature Extraction Scheme to Detect 0-Day Attacks via IDS Alerts," in *International Symposium on Applications and the Internet - SAINT*, Turku, Finlândia, 2008.

[12] P. Kabiri, e A. A. Ghorbani, "Research on Intrusion Detection and Response: A Survey," *International Journal of Network Security*, vol. 2, n. 1, 2005, pp. 84-102.

[13] M. T. Qassrawi and Z. Hongli, "Deception Methodology in Virtual Honeypots," in *Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, Wuhan, China, 2010, pp. 462-467.

[14] L. Spitzner, "Honeypots: Definitions and Value of Honeypots," in *SANS Annual Conference*, 2002.

[15] E. S. Pilli, R.C. Joshi, and R. Niyogi, "Network forensic frameworks: Survey and research challenges," *Digital Investigation*, Elsevier, 2010, pp. 1-14.

[16] S. Atay, and M. Masera, "Challenges for the security analysis of Next Generation Networks," in *Proceedings of the Sixth International Conference on Broadband Communications, Networks, and Systems – BROADNETS*, 2009.

[17] K. Wang, J. Wang, L. Shen, and Z. Han, "An Intelligent Security Defensive Software Scheme and Realization," in *Third International Symposium on Intelligent Information Technology and Security Informatics - IITSI*, 2010, pp. 793-796.

[18] M. Feily, A. Shahrestani, and S. Ramadass, "A Survey of Botnet and Botnet Detection," in *Third International Conference on Emerging Security Information, Systems and Technologies*, 2009.

[19] Act-Net Consortium, Corporate, "The active database management system manifesto: a rulebase of ADBMS features," *ACM SIGMOD Record*, New York, NY, USA, vol. 25, n. 3, 1996, pp. 40-49.

[20] S. Poslad, "Autonomous Systems and Artificial Life," Ubiquitous Computing: Smart Devices, Environments and Interactions, United Kingdom: John Wiley & Sons, Ltd., 2009, pp. 317-341.

[21] N. Agoulmine, S. Balasubramaniam, D. Botvich, J. Strassner, E. Lehtihet, and W. Donnelly, "Challenges for Autonomic Network Management," in *1st IEEE International Workshop on Modelling Autonomic Communications Environments – MACE*, 2006.

[22] Y. Dai, M. Hinchey, M. Qi, and X. Zou, "Autonomic Security and Self-Protection based on Feature-Recognition with Virtual Neurons," in *Proceedings of the 2nd IEEE International Symposium on Dependable, Autonomic and Secure Computing*, 2006.

[23] H. Chen, Y. B. Al-Nashif, G. Qu, and S. Hariri, "Self-Configuration of Network Security," in *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference*, 2007, pp. 97-108.

[24] S. Hariri, L. Xue, H. Chen, M. Zhang, S. Pavuluri, and S. Rao, "AUTONOMIA: an autonomic computing environment," in *Proceedings of the IEEE International Performance, Computing, and Communications Conference*, 2003, pp. 61-68.

[25] G. Qu, S. Hariri, S. Jangiti, J. Rudraraju, S. Oh, S. Fayssal, G. Zhang, and M. Parashar, "Online Monitoring and Analysis for Self-Protection against Network Attacks," in *Proceedings of the International Conference on Autonomic Computing*, 2004, pp. 324-325.

[26] S. Hariri, G. Qu, R. Modukuri, H. Chen, and M. Yousif, "Quality-of-Protection (QoP) - An Online Monitoring and Self-Protection Mechanism," *IEEE Journal on Selected Areas in Communications*, vol. 23, n. 10, 2005, pp. 1983-1993.

[27] B. Wang, P. Zhu, Q. Wen, and X. Yu, "A Honeynet-based Firewall Scheme with Initiative Security Strategies," in *International Symposium on Computer Network and Multimedia Technology - CNMT*, 2009, pp. 1-4.

[28] C. Kreibich, and J. Crowcroft, "Honeycomb - Creating Intrusion Detection Signatures Using Honeypots," *SIGCOMM Computer Communication Review*, ACM, vol. 34, ed. 1, 2004.

[29] N. Provos, "A virtual honeypot framework," in *Proceedings of the 13th conference on USENIX Security Symposium*, vol. 13, Berkeley, CA, USA, 2004.

[30] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time," *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 31, n. 23-24, 1999, pp. 2435-2463.

[31] M. Roesch, "Snort: Lightweight Intrusion Detection for Networks," in *Proceedings of the 13th Conference on Systems Administration*, 1999, pp. 229-238.

[32] G. Portokalidis, and H. Bos, "SweetBait: Zero-hour worm detection and containment using low- and high-interaction honeypots," *Computer Networks*, Elsevier, vol. 51, ed. 5, 2007, pp. 1256-1274.

[33] J. E. Tapiador, and J. A. Clark, "Learning Autonomic Security Reconfiguration Policies," in *IEEE 10th International Conference on Computer and Information Technology (CIT)*, 2010, pp. 902-909.