WIRELESS MONITORING AND CONTROLLING SYSTEM USING CYBER SENSOR NETWORK FOR INDUSTRIAL APPLICATIONS

C.MARAGATHAM, DR.C.N.MARIMUTHU

Abstract— **The expanding reach and wide deployment of digital systems into our most fundamental physical infrastructures pose both a complexity and security concern. The complexity of modern industrial networks primarily stems from three areas: presence of heterogeneous hardware and software, dynamic network composition and usage patterns, and decentralization of control. Dealing with these complexities requires a solution that is flexible and manageable.In existing system, industrial data's are only monitoring through WAN using SOAP and IF-MAP protocol in software manner. But data monitoring is not enough because in critical time admin should have rights to control the industrial data's to avoid the accidents and production wastage.In proposed system, we are using both hardware and software manner. Here SOAP protocol is used in hardware part detects the sensor values through WSN system. IF-MAP is used in software part for data sharing and security purpose for WAN. D-bus is used to link the hardware and software parts. Using this system we can monitor and control the entire industrial system. Here mainly industrial parameters are controlled through WAN.**

I. INTRODUCTION

The expanding reach and wide deployment of digital systems into our most fundamental physical infrastruc-tures pose both a complexity and security concern [1], [2].The complexity ofmodern industrial networks primarily stems from three areas: presence of heterogeneous hardware and soft-ware, dynamic network composition and usage patterns, and decentralization of control [3]. Dealing with these complexities requires a solution that is flexible and manageable.

An autonomic digital ecosystem (DE) is a model for future production systems that builds on the notion of autonomic self-management by embedding exploitable control features within modules [4]. Cyber-physical ecosystems (CPEs) in-clude interconnected subsystems that sense and act upon the physical world to form a complex system of systems. These CPEs are a foundational layer of a wider more encompassing digital ecosystem. CPEs typically bridge the cyber-world of computing and communications with the physical world by embedding wireless sensor nodes into the physical world [5]. Therefore, wireless sensor networks (WSNs) are a fundamental building block of digital ecosystems.

B. IF-MAP Introduction

As was mentioned in the previous section, the communication interface is a critical component of AICS. A messaging inter-face based on the interface to metadata access points (IF-MAP) version 2.0 standard was implemented. IF-MAP is an open pro-tocol standard published by the Trusted Computing Group [15]. Originally made available in April 2008, version 2.0 rev. 47 was published in November 2011. It utilizes a publish, subscribe, messaging paradigm implemented with the simple object ac-cess protocol (SOAP). The design goal of the IF-MAP working group was to enable the sharing of data between network de-vices and systems.

IF-MAP has multiple defined parts: a base protocol and meta-data for a specific usage. Currently, the only metadata defined is for network security, which is sufficient for use in AICS [16]. One of the benefits provided by this definition is a common ref-erence standard of network security data. In addition to the cur-rently defined metadata, it is possible to create a new metadata specification if the base protocol is followed.

1) Base Protocol and Metadata: The base protocol spec-ifies possible actions for clients. IF-MAP clients have access to three actions for metadata: publish, search, and subscribe. Clients store or publish metadata into a metadata access point (MAP) for others to consume. A search allows clients to ob-tain published metadata from the MAP based on a flexible crite-rion. Subscribe requests asynchronous results from a predefined search whenever a client publishes new metadata. In this way, clients can monitor for specific relevant events and be alerted when one occurs. It should be noted that all IF-MAP operations and data types are expressed utilizing XML. This helps ensure a consistent format of the data when it is exchanged between multiple diverse parties.

The protocol also defines two species of data: metadata and identifiers. Metadata in this context is any shared data about net-work devices, policies, status, behavior, or observed relation-ships. The five defined identifiers are identity, IP Address (v4 and v6), MAC address, access request, and device. Identifiers are used to refer to specific metadata items. The metadata of in-terest to this project are as follows.

- **ip-mac**: a binding between an IP address and a mac address valid for a time frame;
- **device-characteristic**: an inherent characteristic of an en-tity such as its manufacturer or OS;

Manuscript received March 24, 2015
C.MARAGATHAM, PG SCHOLAR NANDHA ENGINEERING COLLEGE, ERODE
DR.C.N.MARIMUTHU, PROFESSOR & DEAN/ECE, NANDHA ENGINEERING COLLEGE, ERODE

- **device-ip**: the IP address of an associated device;
- **discovered-by**: a link identifying which sensor device has discovered a new IP or MAC address;
- **event**: activity of interest on a network such as a virus attack or abnormal behavior.

2) SOAP: All IF-MAP actions are transmitted using SOAP. The SOAP version 1.2 protocol is a specification utilizing XML for exchanging structured and typed information in a distributed environment. Typically, hypertext transfer protocol (HTTP) or SMTP is used for message handling. It provides an extensible framework for transferring application specific information without specifically detailing the semantics of the data carried. The framework provides required actions to perform on a SOAP message in order to send and receive it. SOAP is typically used as a messaging framework layer of a Web-based SOA. SOA is an enabling technology for develop-ment of large systems in industry.

C. D-Bus

In this paper, the use of D-Bus is described for internal com-munication between sensor components and the external mes-saging service. D-Bus is a system created for interprocess com-munication (IPC) that has been used in reconfigurable mobile network devices [18]. It was designed to avoid round trips and allow for asynchronous operation. Conceptually, it works in terms of messages and handles many of the difficult IPC syn-chronization issues. Several high-level language bindings are available, which provides a flexible implementation solution. The following describes the D-Bus system and the terminology commonly used with it.

A *bus* is a virtual path that messages are passed over. Mul-tiple instances of a bus may exist and are managed by a soft-ware daemon. Several Linux-based operating systems offer two busses, system and session, for use without requiring user con-figuration. Fine-grained user access to buses and actions on the appropriate bus service is available via configuration files. Ap-plications that utilize a bus take on the role of either a server or client. Server processes listen for incoming method requests from clients.

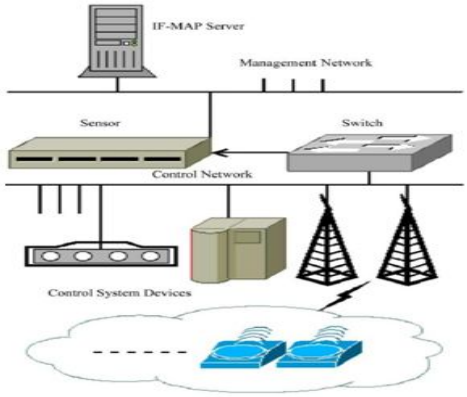
A *method* is made available by publishing it on a message bus. Methods are remotely invoked operations of an object. As is found in object-oriented programming languages, objects con-tain methods. The input and output parameters need to be de-fined in advance of publication. A method is registered with the D-Bus daemon under an *interface* and available through an *ob-ject path*. Interfaces are a collective group of methods that help define the type of an object.

Fuzzy logic provides a framework for system modeling in lin-guistic form capable of coping with imprecise and vague mean-ings of words. The Type-2 Fuzzy Logic System (T2 FLS) is an extension of the Type-1 Fuzzy Logic System (T1 FLS) that has been successful in modeling and minimizing the effects of var-ious kinds of dynamic uncertainties. Here, the Interval Type-2 Fuzzy Logic System (IT2 FLS), as a specific case of T1 FLS, is used to encode the linguistic domain knowledge about the spe-cific network system and dynamically adjust the sensitivity of the anomaly detection algorithms.

The IAA component is based on an existing low-cost online rule extraction technique [21]. The model is composed of a set of fuzzy rules that are constructed based on a window-based feature vector using an online version of the adapted nearest neighbor clustering (NNC) algorithm. The anomaly detection algorithm was specifically designed to allow for both fast learning and fast classification on the constrained computa-tional resources of an embedded device. The overall architecture of the proposed anomaly detection system is depicted in Fig. 3. The network traffic is processed by an IT2 FLS that uses a fuzzy logic rule base with encoded lin-guistic domain knowledge to calculate the cyber-security con-text. This cyber-security context expresses the belief that an in-truder is currently present in the system.

In the next stage, the network traffic is separated into indi-vidual communication streams. Each stream is delineated by an IP address retrieved from the internal message bus. Other features, such as port numbers or protocol types, can also be used. Packets assigned to individual communication streams are then passed into dedicated anomaly detection algorithms. Each anomaly detection algorithm maintains its own buffer of in-coming packets, which is used to extract a set of window-based features.

AICS was developed as a deployable modular framework on a commonly available hardware platform with components that provide network host information, identify anomalous network traffic behaviors and deploy dynamic virtual honeypots. In the test scenario, the sensor was deployed on a self-contained hard-ware platform. This effectively hides the communication be-tween components from other network entities, thereby adding another layer of defense. When considering the possible addi-tion of information from external hosts on the IF-MAP interface, a balance between security and information access would need to be evaluated.



II. COMPARISON TO EXISTING WORK AND EVALUATION

A related anomaly detection project by Dussel *et al.* relies on the computation of similarity between transport-layer packet payloads embedded in a geometric space [28]. They performed experiments on traffic from a SCADA test bed containing various application-layer protocols. A primary difference from AICS includes the use of packet data, as opposed to header derived information, to produce n-gram populated feature vectors. A simple distance measure is then compared with stored "normal" vectors to determine if an anomaly exists in the traffic. The AICS IAA algorithm maintains two models of system behavior: a global threat model that considers all traffic and a set of individual models for a given stream. Their solution appears to broadly apply itself to all traffic. It is difficult to compare results with

different test data sets, but their average detection rate of 88%–92% at a false positive level of 0.2% may be due to the differences just mentioned.

An additional beneficial outcome of the presented solution is removal of the configuration burden from the human oper-ator. This capability stems from the self-configuring aspects of an Autonomic design. For example, the IAA anomaly detection routine doesn't require human created rules. It learns normal be-havior from observations of the system. In addition, the dynamic honeypots are configured from system observations. This func-tionality reduces dependence on network expertise and level of human configuration effort.

The CPU and RAM utilization was monitored during the test scenario. As the hardware platform is a fanless system, temper-ature data was recorded as well. The 200 000 anomaly evalu-ation packets contained 19 627 063 bytes of payload informa-tion. Approximately 97 packets per second were processed normal network transmission speed. It was observed that each anomaly alert sent from IAA on the IF-MAP mechanism in-curred an overhead of .7 ms of communication processing time.

The CPU thermal sensors were polled every 5 s. Room tem-perature at the start of the test was 23 ◦C. Core 0 temperature after a 5-min warm-up period was 44 ◦C. Values during the test ranged from 42 ◦C to 47 ◦C. Core 1 temperature during the warm-up period was 45 ◦C. The test values ranged from 43 •C to 48 ◦C. It is speculated that the difference in the range low value and the initial start might be due to variations in room temperature during the test.

The at-rest CPU utilization, as measured by the Linux top command on 1-s intervals, was 1.1%. This was due mainly to the Xorg process providing HMI services. The utilization max-imum during the test was 47.4% with an average of .65% and standard deviation of 2.29%. The maximum occurred when the virtual honeypots were probed with Nmap. The graph in Fig. 7 shows the sampled CPU utilization before and after the max-imum that occurred between time index 100 and 181. Another key performance indicator for Linux machines is the amount of time the CPU has been waiting for I/O to complete. The max-imum value was 10% with an average of 0.59% and standard deviation of 0.62%. In other words, the machine was not over-whelmed with the test data when running in real time.

CONCLUSION

Inspired by self-configuring aspects of autonomic computing, the work presented here supports the important goal of securing industrial ecosystems by providing network security awareness in a heterogeneous control system network. Contributions of this paper include 1) a flexible two-level communication layer based on autonomic computing and service-oriented architec-ture and 2) descriptions of three dynamic modules that respond to a changing environment.

A novel working sensor prototype was developed based on a unique implementation of the Trusted Network Group's IF-MAP Web-services-based communication protocol. Sensor communication between self-contained modules is accom-plished with D-Bus. At multiple steps of a test scenario, the communication layer was utilized to provide information in a well-defined format between components and to external entities.

As can be seen in the architecture and scenario sections, the self-configuration capability of autonomic systems is exempli-fied by the IAA and DHP components. The DHP host entities are created automatically based upon passive monitoring of net-work activity. Anomaly behavior in IAA is detected by clus-tering of information retrieved from a moving window of traffic.

The only source of outside information explicitly provided is the IP addresses of devices deemed to be critical in the functioning system. This information is asynchronously delivered by a de-coupled two-part communication system.

Multiple intelligent modules were deployed on a test system to monitor for anomalous behavior and create deceptive em-ulated hosts. The modules are exemplary implementations of self-learning components. In a test scenario, 45 of the 46 net-work attached devices were recognized, and 10 of the 12 em-ulated devices were created with representative characteristics. In addition, 99.9% of anomalous packets were recognized. The modules utilized the communication layer to provide notifica-tions and retrieve information.

Future work includes integration of AICS with external com-ponents utilizing the IF-MAP standard. These components in-clude functionality to correlate information from the sensor with production data. Multiple sensors will be deployed necessitating further use Web standards such as WS-Management.

REFERENCES

[1] V. C. Gungor and G. P. Hancke, "Industrial wireless sensor networks: Challenges, design principles, and technical approaches," *IEEE Trans. Ind. Electron.*, vol. 56, no. 10, pp. 4258–4265, Oct. 2009.

[2] M. Cheminod, L. Durante, and A. Valenzano, "Review of security is-sues in industrial networks," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 277–293, 2013.

[3] V. C. Gungor, B. Lu, and G. P. Hancke, "Opportunities and challenges of wireless sensor networks in smart grid," *IEEE Trans. Ind. Electron*, vol. 57, no. 10, pp. 3557–3564, Oct. 2010.

[4] M. Ulieru and S. Grobbelaar, "Engineering industrial ecosystems in a networked world," in *Proc. 5th IEEE Int. Conf. Ind. Inf.*, June 23–27, 2007, vol. 1, pp. 1–7.

[5] Y. Yang, Y. Xu, X. Li, and C. Chen, "A loss inference algorithm for wirelss sensor networks to improve data reliability of digital ecosys-tems," *IEEE Trans. Ind. Electron*, vol. 58, no. 6, pp. 2126–2137, Jun. 2011.

[6] H. Ruan, M. Lacoste, and J. Leneutre, "A policy management frame-work for self-protection of pervasive systems," in *Proc. 6th Int. Conf. Autonom. Autonom. Syst.*, Mar. 2010, pp. 104–109.

[7] G. Candido, A. W. Colombo, J. Barata, and F. Jammes, "Service-ori-ented infrastructure to support the deployment of evolvable production systems," *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 759–767, Nov. 2011.

[8] R. Kyusakov, J. Eliasson, J. Delsing, J. Deventer, and J. Gustafsson, "Integration of wireless sensor and actuator nodes with IT infrastruc-ture using service-oriented architecture," *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 43–51, 2013.

[9] R. Agrawal, W. Grosky, and F. Fotouhi, "Virus detection and removal service architecture in digital ecosystems," in

Proc. IEEE Conf. Digit. Ecosyst. Tech., Jun. 2009, pp. 301–305.

[10] S. Dobson *et al.*, "A survey of autonomic communications," *ACM Trans. Autonom. Adapt. Syst.*, vol. 1, no. 2, pp. 223–259, Dec. 2006.

[11] P. Brodal, *The Central Nervous System*. Oxford, U.K.: Oxford Univ. Press, Jan. 15, 1998.

[12] IBM, An architectural blueprint for autonomic computing," white paper, 4th ed., Jun. 2006 [Online]. Available: http://www-03.ibm.com/ autonomic/pdfs/ACBlueprintWhitePaper4th.pdf

[13] Y. Cheng, A. Leon-Garcia, and I. Foster, "Toward an autonomic ser-vice management framework: A holistic vision of SOA, AON, and au-tonomic computing," *IEEE Comm. Mag.*, vol. 46, no. 5, pp. 138–146, 2008.

[14] H. Shuaib, R. J. Anthony, and M. Pelc, "Towards certifiable autonomic computing systems," Autonomic Research Group, CMS, University of Greenwich, Greenwich, U.K., TR 1, 2010.

[15] TNC IF-MAP Binding for SOAP, Trusted Computing Group. ver. 2.0, Nov. 2011 [Online]. Available: http://www.trustedcomputing-group.org/resources/tnc_if map_binding_for_soap_specification

[16] TNC IF-MAP Metadata for Network Security. ver. 1.0, Nov. 2010 [Online]. Available: http://www.trustedcomputinggroup.org/re-sources/tnc_if map_metadata_for_network_security