International Journal of Information Technology and Management

Vol. X, Issue No. XV, May-2016, ISSN 2249-4510

INTRUSION DETECTION OF MEDICAL DEVICES EMBEDDED IN A MEDICAL CYBER PHYSICAL SYSTEM

AN INTERNATIONALLY INDEXED PEER REVIEWED & REFEREED JOURNAL

www.ignited.in

International Journal of Information Technology and Management
Vol. X, Issue No. XV, May-2016, ISSN 2249-4510

Intrusion Detection of Medical Devices Embedded in a Medical Cyber Physical System

Arshad Shareef

Assistant Professor, Department of CSE, Arkay College of Engineering & Technology, Bodhan, Dist. Nizamabad, Telangana State

Abstract – This paper proposes and analyzes a behavior-rule specification-based technique for intrusion detection of medical devices embedded in a medical cyber physical system (MCPS) in which the patient's safety is of the utmost importance. A methodology is proposed to transform behavior rules to a state machine, so that a device that is being monitored for its behavior can easily be checked against the transformed state machine for deviation from its behavior specification. Using vital sign monitor medical devices as an example; to demonstrate the intrusion detection technique that can effectively trade false positives off for a high detection probability to cope with more sophisticated and hidden attackers to support ultra-safe and secure MCPS applications. Moreover, through a comparative analysis, it is demonstrated that the proposed behavior-rule specification-based IDS technique outperforms two existing anomaly-based techniques for detecting abnormal patient behaviors in pervasive healthcare applications.

Keywords: Intrusion Detection, Sensor Actuator Networks, Medical Cyber Physical Systems, Healthcare, Security, Safety.

----------X----------

INTRODUCTION

Internet services and applications have become an inextricable part of daily life, enabling communication and the management of personal information from anywhere. To accommodate this increase in application and data complexity, web services have moved to a multi-tiered design wherein the web server runs the application front-end logic and data are outsourced to a database or file server.

Double Guard differs from this type of approach that correlates alerts from independent IDSs. Rather, Double-Guard operates on multiple feeds of network traffic using single IDS that looks across sessions to produce an alert without correlating or summarizing the alerts produced by other independent IDSs. This system used to detect attacks in multi-tiered web services. This approach can create normality models of isolated user sessions that include both the web front-end (HTTP) and back-end (File or SQL) network transactions. For websites that do not permit content modification from users, there is a direct causal relationship between the requests received by the front-end web server and those generated for the database back end.

No prior knowledge of the source code or the application logic of web services deployed on the web server. Virtualization is used to isolate objects and enhance security performance. Lightweight containers can have considerable performance advantages over full virtualization

Existing System:

- Firewall program used
- Degrade system performance
- Slower process
- Algorithm not efficient

Disadvantage:

- This container-based and session-separated web server architecture not only enhances the security performances.
- Privilege Escalation Attack
- Vulnerabilities Due to Improper Input Processing
- Distributed DoS

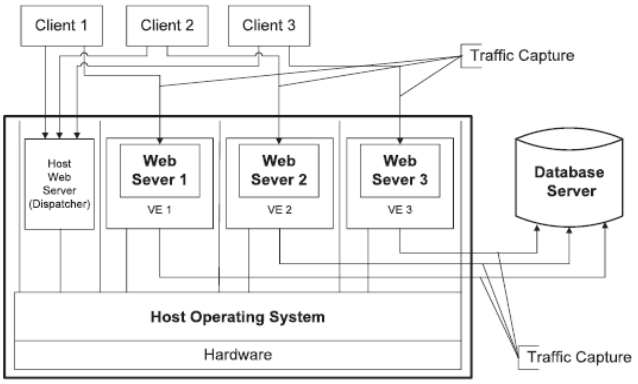
Proposed System:

- Static Model building algorithm used.
- It employs a lightweight virtualization technique to assign each user's web session to a dedicated container, an isolated virtual computing environment.
- It uses the container ID to accurately associate the web request with the subsequent DB queries. Thus, Double Guard can build a causal mapping profile by taking both the web server and DB traffic into account.
- In addition to this static website case, there are web services that permit persistent back-end data modifications. These services, which we call dynamic, allow HTTP requests to include parameters that are variable and depend on user input.

Advantages:

- Double Guard was able to identify a wide range of attacks with minimal false positives.
- Easy to find the Direct DB Attack.
- Light weight virtualization technique

Architecture:



MODULES

List of Modules

- Session Monitoring
- Analysis of Dataset
- Static Model Building Algorithm
- Attack Detection

Session Monitoring:

This prototype chose to assign each user session into a different container; however, this was a design decision. For instance, a new session per each new IP address of the client can be assigned. In this implementation, sessions were recycled based on events or when sessions time out. The same session can be used for tracking mechanisms as implemented by the Apache server because lightweight virtualization containers do not impose high memory and storage overhead. Thus, a large number of parallel-running Apache instances similar to the Apache threads that the server would maintain in the scenario without session containers. If a session timed out, the Apache instance was terminated along with its container. This prototype implementation uses a 60-minute timeout due to resource constraints of the test server.

Analysis of Dataset:

Based on the web server's application logic, different inputs would cause different database queries. For example, to post a comment to a blog article, the web server would first query the database to see the existing comments.

If the user's comment differs from previous comments, then the web server would automatically generate a set of new queries to insert the new post into the back-end database. Otherwise, the web server would reject the input in order to prevent duplicated comments from being posted (i.e., no corresponding SQL query would be issued). In such cases, even assigning the same parameter values would cause different set of queries, depending on the previous state of the website.

Likewise, this nondeterministic mapping case (i.e., one-to-many mapping) happens even after we normalize all parameter values to extract the structures of the web requests and queries. Since the mapping can appear differently in different cases, it becomes difficult to identify all of the one-to-many mapping patterns for each web request. Moreover, when different operations occasionally overlap at their possible query set, it becomes even harder to extract the one-to-many mapping for each operation by comparing matched requests and queries across the sessions.

Static Model Building Algorithm:

An algorithm is developed that takes the input of training data set and builds the mapping model for websites. For each unique HTTP request and database query, the algorithm assigns a hash table entry, the key of the entry is the request or query itself, and the value of the hash entry is AR for the request or AQ for the query, respectively. The algorithm generates the mapping model by

considering all mapping patterns that would happen in websites.

Attack Detection:

The attacker visits the website as a normal user aiming to compromise the web server process or exploit vulnerabilities to bypass authentication. At that point, the attacker issues a set of privileged (e.g., admin-level) DB queries to retrieve sensitive information. Log and process both legitimate web requests and database queries in the session traffic, but there are no mappings among them. Double Guard separates the traffic by sessions.

If it is a user session, then the requests and queries should all belong to normal users and match structurally. Using the mapping model that is created during the training phase, Double Guard can capture the unmatched cases. The mappings are being established between HTTP requests and database queries, clearly defining which requests should trigger which queries. For an SQL injection attack to be successful, it must change the structure (or the semantics) of the query, which this approach can readily detect.

First of all, according to the proposed mapping model, DB queries will not have any matching web requests during this type of attack. On the other hand, as this traffic will not go through any containers, it will be captured as it appears to differ from the legitimate traffic that goes through the containers. Double Guard is designed to mitigate DDoS attacks. These attacks can occur in the server architecture without the back-end database.

Attack Scenarios

The Proposed system is effective at capturing the following types of attacks:

- Privilege Escalation Attack
- Hijack Future Session Attack
- Injection Attack
- Direct DB Attack

Privilege Escalation Attack:

Let's assume that the website serves both regular users and administrators. For a regular user, the web request ru will trigger the set of SQL queries Qu; for an administrator, the request ra will trigger the set of admin level queries Qa. Now suppose that an attacker logs into the web server as a normal user, upgrades his/her privileges, and triggers admin queries so as to obtain an administrator's data. This attack can never be detected by either the web server IDS or the database IDS since both ru and Qa are legitimate requests and queries. This approach, however, can detect this type of attack since the DB query Qa does not match the request ru, according to the said mapping model.

Hijack Future Session Attack:

This class of attacks is mainly aimed at the web server side. An attacker usually takes over the web server and therefore hijacks all subsequent legitimate user sessions to launch attacks. For instance, by hijacking other user sessions, the attacker can eavesdrop, send spoofed replies, and/or drop user requests. A session-hijacking attack can be further categorized as a Spoofing/Man-in-the-Middle attack, an Exfiltration Attack, a Denial-of-Service/Packet Drop attack, or a Replay attack. According to the mapping model, the web request should invoke some database queries (e.g., a Deterministic Mapping then the abnormal situation can be detected. However, neither a conventional web server IDS nor a database IDS can detect such an attack by itself. Fortunately, the isolation property of the container based web server architecture can also prevent this type of attack. As each user's web requests are isolated into a separate container, an attacker can never break into other users' sessions.

Injection Attack:

Attacks such as SQL injection do not require compromising the web server. Attackers can use existing vulnerabilities in the web server logic to inject the data or string content that contains the exploits and then use the web server to relay these exploits to attack the back-end database. Since this approach provides two-tier detection, even if the exploits are accepted by the web server, the relayed contents to the DB server would not be able to take on the expected structure for the given web server request. For instance, since the SQL injection attack changes the structure of the SQL queries, even if the injected data were to go through the web server side, it would generate SQL queries in a different structure that could be detected as a deviation from the SQL query structure that would normally follow such a web request

Direct DB Attack:

It is possible for an attacker to bypass the web server or firewalls and connect directly to the database. An attacker could also have already taken over the web server and be submitting such queries from the web server without sending web requests. Without matched web requests for such queries, a web server IDS could detect neither. Furthermore, if these DB queries were within the set of allowed queries, then

the database IDS it would not detect it either. However, this type of attack can be caught with this approach since one cannot match any web requests with these queries

Double Guard Limitations:

This section discusses the operational and detection limitations of Double Guard.

Vulnerabilities Due to Improper Input Processing:

Cross Site Scripting is a typical attack method where in attackers embedding malicious client scripts via legitimate user inputs. In Double Guard, the entire user input values are normalized so as to build a mapping model based on the structures of HTTP requests and DB queries. Once the malicious user inputs are normalized, Double Guard cannot detect attacks hidden in the values. These attacks can occur even without the databases. Double Guard offers a complementary approach to those research approaches of detecting web attacks based on the characterization of input values.

Possibility of Evading Double Guard:

The assumption is that an attacker can obtain "full control" of the web server thread that he/she connects to. That is, the attacker can only take over the web server instance running in its isolated container. The proposed architecture ensures that every client be defined by the IP address and port container pair, which is unique for each session. Therefore, hijacking an existing container is not possible because traffic for other sessions is never directed to an occupied container. If this were not the case, the said architecture would have been similar to the conventional one where a single web server runs many different processes. Moreover, if the database authenticates the sessions from the web server, then each container connects to the database using either admin user account or non admin user account and the connection is authenticated by the database.

In such case, an attacker will authenticate using a non admin account and will not be allowed to issue admin level queries. In other words, the HTTP traffic defines the privileges of the session which can be extended to the back-end database, and a non admin user session cannot appear to be an admin session when it comes to back-end traffic. Within the same session that the attacker connects to, it is allowed for the attacker to launch "mimicry" attacks. It is possible for an attacker to discover the mapping patterns by doing code analysis or reverse engineering, and issue "expected" web requests prior to performing malicious database queries.

However, this significantly increases the efforts for the attackers to launch successful attacks. In addition, users with non admin permissions can cause minimal (and sometimes zero) damage to the rest of the system and therefore they have limited incentives to launch such attacks. By default, Double Guard normalizes all the parameters. Of course, the choice of the normalization parameters needs to be performed carefully. Double Guard offers the capability of normalizing the parameters so that the user of Double Guard can choose which values to normalize.

For example, it can chose not to normalize the value "admin" in "user= 'admin'." Likewise, one can choose to normalize it if the administrative queries are structurally different from the normal-user queries, which is common case. Additionally, if the database can authenticate admin and non admin users, then privilege escalation attacks by changing values are not feasible (i.e., there is no session hijacking).

Distributed DoS:

Double Guard is not designed to mitigate DDoS attacks. These attacks can also occur in the server architecture without the back-end database.

Anomaly detection:

Anomaly detection also referred to as **outlier detection** refers to detecting patterns in a given data set that do not conform to an established normal behaviour. The patterns thus detected are called anomalies and often translate to critical and actionable information in several application domains. Anomalies are also referred to as outliers, change, deviation, surprise, aberrant, peculiarity, intrusion, etc.

In particular in the context of abuse and network intrusion detection, the interesting objects are often not *rare* objects, but unexpected bursts in activity. This pattern does not adhere to the common statistical definition of an outlier as a rare object, and many outlier detection methods (in particular unsupervised methods) will fail on such data, unless it has been aggregated appropriately. Instead, a cluster analysis algorithm may be able to detect the micro clusters formed by these patterns.

Three broad categories of anomaly detection techniques exist. **Unsupervised anomaly detection** techniques detect anomalies in an unlabeled test data set under the assumption that the majority of the instances in the data set are normal by looking for instances that seem to fit least to the remainder of the data set.

Supervised anomaly detection techniques require a data set that has been labelled as "normal" and "abnormal" and involves training a classifier (the key difference to many other statistical classification problems is the inherent unbalanced nature of outlier detection). **Semi-supervised anomaly detection** techniques construct a model representing normal behaviour from a given *normal* training data set, and

then testing the likelihood of a test instance to be generated by the learnt model.

CONCLUSION

For safety-critical MCPSs, being able to detect attackers while limiting the false alarm probability to protect the welfare of patients is of utmost importance. This paper proposed a behavior-rule specification-based IDS technique for intrusion detection of medical devices embedded in a MCPS. This paper also exemplified the utility with VSMs and demonstrated that the detection probability of the medical device approaches one (that is, we can always catch the attacker without false negatives) while bounding the false alarm probability to below 5 percent for reckless attackers and below 25 percent for random and opportunistic attackers over a wide range of environment noise levels. Through a comparative analysis, it is demonstrated that the behavior rule specification-based IDS technique outperforms existing techniques based on anomaly intrusion detection.

REFERENCES

G. Chen, W. He, J. Liu, S. Nath, L. Rigas, L. Xiao, and F. Zhao, "Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services," Proc. Fifth USENIX Symp. Networked Systems Design and Implementation (NSDI '08), pp. 337-350, 2008.

J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processingon Large Clusters," in Proc. 6th Symp. Operating System Design and Implementation (OSDI'04), Dec. 2004, pp. 137–150.

J. Hamilton, "Cooperative Expendable Micro-Slice Servers (CEMS): Low Cost, Low Power Servers for Internet-Scale Services," Proc. Fourth Biennial Conf. Innovative Data Systems Research (CIDR), Jan. 2009.

J. Liu, F. Zhao, X. Liu, and W. He, "Challenges towards Elastic Power Management in Internet Data Centers," Proc. 29th IEEE Int'l Conf. Distributed Computing Systems Workshops (ICDCSW '09), pp. 65-72, 2009.

L. Rao, X. Liu, L. Xie, and W. Liu, "Minimizing Electricity Cost: Optimization of Distributed Internet Data Centers in a Multi- Electricity-Market Environment," Proc. IEEE INFOCOM, pp. 1-9, Mar. 2010,

L. Rao, X. Liu, M.D. Ilic, and J. Liu, "Distributed Coordination of Internet Data Centers under Multiregional Electricity Markets," Proc. IEEE, vol. 100, no. 1, pp. 269-282, Jan. 2011.

R. Ge, X. Feng, and K. Cameron, "Performance-Constrained Distributed DVS Scheduling for Scientific Applications on Power-Aware Clusters," Proc. ACM/IEEE SC Conf. Supercomputing, p. 34, Nov. 2005.

R. Nathuji and K. Schwan, "Virtualpower: Coordinated Power Management in Virtualized Enterprise Systems," Proc. 21st ACM SIGOPS Symp. Operating Systems Principles (SOSP '07), pp. 265-278, 2007.

R. Urgaonkar, B. Urgaonkar, M.J. Neely, and A. Sivasubramaniam, "Optimal Power Cost Management Using Stored Energy in Data Centers," Proc. ACM SIGMETRICS Joint Int'l Conf. Measurement and Modeling of Computer Systems (SIGMETRICS '11), pp. 221-232, 2011.

S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," in Proc. 19th ACM Symp. Operating Systems Principles (SOSP'03), Oct. 2003, pp. 29–43.

Y. Guo and Y. Fang, "Electricity Cost Saving Strategy in Data Centers by Using Energy Storage," IEEE Trans. Parallel and Distributed Systems, vol. 24, no. 6, pp. 1149-1160, June 2013.

Z. Liu, M. Lin, A. Wierman, S.H. Low, and L.L. Andrew, "Greening Geographical Load Balancing," ACM SIGMETRICS Performance Evaluation Rev., vol. 39, pp. 193-204, June 2011.