Comparative Analysis of Firmware Security: A Proactive Paradigm for Enhancing Efficiency and Adaptability through Anomaly Detection

Sanjay J[1], Amrutha E[2], Neha Ramesh[3], Vaidehi Vijayakumar[4] (Senior Member IEEE)
SCOPE, *Vellore Institute of Technology,* Chennai, India
[1]sanjay.j2022@vitstudent.ac.in, [2]amrutha.e2022@vitstudent.ac.in, [3]neha.r2022a@vitstudent.ac.in,
[4]vaidehi.vijayakumar@vit.ac.in

2023 6th International Conference on Recent Trends in Advance Computing (ICRTAC) | 979-8-3503-9470-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICRTAC59277.2023.10480848

***Abstract*— Firmware security, which serves as the link between software and hardware operations, arises as a crucial problem in the ever-changing world of computer hardware. Firmware's centrality, however, makes it vulnerable to a wide range of security risks, from unauthorized upgrades to malicious code insertion, with potentially dire implications including data leaks and device malfunction. Even if they are useful, current firmware security analysis techniques have drawbacks when it comes to proactive threat detection and resource intensity. This paper aims to transform firmware security by proposing a novel technique named Behavior-Based Anomaly Detection (BBAD). BBAD employs a proactive approach, utilizing advanced machine learning algorithms and real-time firmware behavior monitoring to identify and eliminate security vulnerabilities prior to their occurrence. BBAD offers improved performance, proactive threat detection, and flexibility to changing security environments, making it a potential path for firmware security in the future. This study offers a thorough investigation that includes a review of relevant literature, an explanation of BBAD, a comparative analysis, a conclusion, and suggestions for further research.**

Keywords—Firmware security, Threats, Behavioral Based Anomaly detection

I. Introduction

Firmware security is an essential part of today's computer hardware, acting as a bridge between the hardware and software worlds. The firmware at the heart of these devices plays a key role in device performance and data integrity. But it is this centrality that makes firmware an attractive target for many security threats. From injecting malicious code to unauthorized firmware updates [1,4] and exploiting security holes, there are many potential risks associated with firmware. The consequences of a firmware breach can be serious, ranging from unauthorized use and data breaches to complete disruption of device functionality.

Current firmware security analysis methods like static analysis and dynamic analysis have their limitations [2]. Although valuable in their own right, they often require significant computing resources and may not be able to proactively detect new threats. In addition, their effectiveness, they can be weakened by rapidly evolving cyber security threats[12].

Hence this research study suggests a novel approach named Behaviour Based Anomaly Detection (BBAD) to deal with these issues. The proposed method has a proactive paradigm ability, to detect security threats before they occur contributing to a proactive security stance. BBAD's adaptiblity through anomaly detection ensures adaptability to changing security landscapes. It makes use of real-time firmware behaviour monitoring to build baseline profiles and identify irregularities that might be signs of security risks. These abnormalities are continually analysed by sophisticated machine learning algorithms and a feedback loop, allowing for quick mitigation actions. The proposed method promises to transform firmware security evaluation by providing enhanced performance, proactive threat detection, and flexibility.

This paper is organised as follows: Section II overviews the related literature, Section III presents the proposed Behaviour Based Anomaly Detection (BBAD), Section IV gives the comparative analysis, Section V presents the conclusion and Section VI presents the future work.

II. Related Literature

A. Static analysis

Static analysis is a basic approach to firmware security that involves examining firmware code without actually running it. The purpose of this method is to identify vulnerabilities, suspicious code patterns and potential security issues by directly analyzing the firmware image. Static analyzers look at the source code or binary representation of the firmware and look for known vulnerabilities, such as hard-coded credentials, buffer overflows, or insecure function calls. This approach is valuable for identifying vulnerabilities in the firmware code base regardless of the execution path, making it effective for detecting identified security risks [15]. However, static analysis may have limitations in detecting vulnerabilities in a silent or rarely executed code path, and may have difficulty identifying new or zero-day vulnerabilities that are not part of its predefined signature database. However, static analysis, when combined with other security assessment techniques, is critical to fortifying firmware against known threats and providing basic protection for modern computing devices.

B. Dynamic analysis

Dynamic analysis is a key method of firmware security that involves running firmware code in a controlled

environment to observe its behavior in real time. The goal of this approach is to uncover vulnerabilities, malicious activities, or unexpected behaviors that may not be apparent from static analysis alone. During dynamic analysis [14], various inputs and scenarios are applied to the firmware to evaluate how it responds. This process helps identify runtime vulnerabilities, memory leaks and interactions with external systems. Dynamic analysis is particularly effective for finding previously unknown vulnerabilities and firmware behavior under different conditions. However, it may not be comprehensive as static analysis in identifying coded or design-related vulnerabilities. By combining dynamic analysis with other security assessment techniques, such as static analysis and fuzz testing, a more comprehensive understanding of firmware security can be achieved, making ita valuable part of the overall security strategy for modern computing devices.

C. *Fuzz testing*

Fuzz testing is a dynamic analysis technique used to identify vulnerabilities in firmware security by exposing the firmware to multiple unexpected or malformed inputs. The goal of this approach is to identify input validation weaknesses, buffer overflows, or other code execution vulnerabilities that can be exploited by malicious actors. In Fig. 1, during fuzz testing, automated tools generate a variety of input data, such as random or intentionally malformed data packets, and send them to hardware input interfaces. By doing so, fuzz testing tends to cause unexpected functionality, crashes or firmware security bugs [3]. When vulnerabilities are triggered, they can be detected and analyzed for possible exploitation.

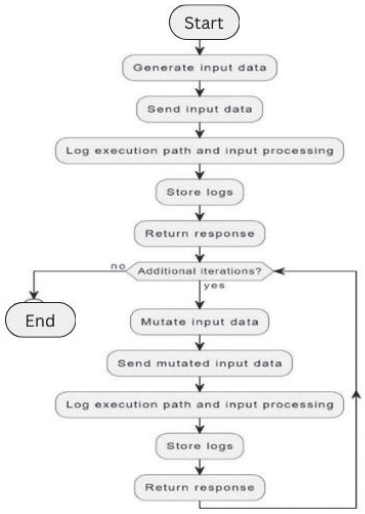

Fig. 1. Fuzz testing

Fuzz testing is very effective at detecting input-related vulnerabilities and is especially valuable when evaluating how firmware handles unexpected or malicious inputs [8]. It complements other assessment techniques such as static and binary analysis, providing a holistic approach to firmware security that helps developers and security experts identify and fix vulnerabilities, improving the overall resilience of computing devices [7].

D. *Binary analysis*

Binary analysis is a critical approach to firmware security that involves extracting firmware binary code, usually without access to the original source code. The purpose of this method is to understand the functionality of the firmware, identify vulnerabilities and analyze how it interacts with hardware and external systems. By recovering the binary representation of firmware, security researchers can uncover hidden vulnerabilities, study firmware behavior, and assess potential security risks. Binary analysis is necessary when dealing with proprietary firmware where access to the source code is limited or absent [6]. It helps identify vulnerabilities that may not be visible with other assessment methods and provides information about how firmware interacts with hardware components. However, this can be a complex and time-consuming process that requires special skills and tools. When binary analysis is combined with static and dynamic analysis, it provides a complete picture of firmware security and helps develop effective countermeasures against potential threats and vulnerabilities. Table I presents the comparison of various security methods namely: static, dynamic, binary and fuzz.

TABLE I. COMPARISON OF VARIOUS SECURITY METHODS

Approach	***Objective***	***Limitation***	***Evaluation metrics***
Static Analysis	Identify vulnerabilities, hard coded credentials, and code patterns without execution.	May miss Vulnerabilities in dormant code paths; Limited Effectiveness against novel or zero-day threats.	Vulnerabilities Detected, False Positives, Code Complexity
Dynamic Analysis	Analyze firmware behavior during runtime to detect vulnerabilities and unexpected behaviors.	Resource intensive; May not identify vulnerabilities in dormant code; Limited to observable behaviors.	Anomalies Detected, ExecutionTime, Resource Usage
Binary Analysis	Dissect firmware Binaries to understand functionality interactions, and hidden vulnerabilities.	Limited effectiveness for proprietary firmware; Requires reverse engineering skills; Resource intensive.	Hidden Vulnerabilities, Discovered Function Call Analysis, Code Complexity
Fuzz Testing	Discover vulnerabilities Related to input validation, buffer overflows and unexpected inputs.	Limited to input- related vulnerabilities; May not find Complex logic errors; Requires input mutation	Vulnerabilities Discovered, Code Coverage, Input Mutation

III. BEHAVIOR BASED ANOMALY DETECTION (BBAD)

The proposed Behavioral Based Anomaly Detection (BBAD) approach is an innovative approach to firmware

security analysis. The sequence diagram of BBAD is shown in Fig.2. Unlike traditional methods, BBAD focuses on real-time monitoring of the behavior of firmware components during execution. It starts by creating base-level behavioral profiles for each firmware element, taking into account aspects such as system calls, memory usage and hardware interactions. Advanced machine learning models, including anomaly detection algorithms and neural networks, continuously analyze firmware behavior and look for deviations from established baselines. Importantly, BBAD includes feedback that adapts its core profile over time to evolving firmware features. When anomalies are detected that indicate potential security threats, BBAD triggers real-time alerts that enable immediate mitigation measures.

A. BBAD Algorithm

```
def instrument_firmware(firmware):
    target_firmware= load_firmware(firmware)
    instrumented_firmware = instrument(target_firmware)
    return instrumented_firmware
def collect_data(instrumented_firmware):
    behavioral_data =
    run_instrumented_firmware(instrumented_firmware)
    store_data(behavioral_data)
def train_behavioral_profiles(historical_data):
    machine_learning_models =
    train_models(historical_data)
    return machine_learning_models
def integrate_with_firmware(firmware,
  machine_learning_models):
  secure_firmware =integrate(firmware,
  machine_learning_models)return secure_firmware
def monitor_behavior(secure_firmware):
    while True:
    current_behavior = monitor(secure_firmware)
    anomalies = detect_anomalies(current_behavior,
    machine_learning_models)
    if anomalies: generate_alerts(anomalies)
def mitigate_anomalies(alerts):
    for alert in alerts: analyze_and_mitigate(alert)
def update_behavioral_profiles(historical_data,
    machine_learning_models):
    new_profiles =retrain_models(historical_data)
    machine_learning_models =
    update_models(machine_learning_models,
    new_profiles)
def run_bbad_algorithm():
    while True: monitor_behavior(secure_firmware)
    update_behavioral_profiles(historical_data,
    machine_learning_models)
def evaluate_effectiveness():
    risk_analysis = analyze_risk(secure_firmware)
    case_studies = conduct_case_studies(secure_firmware)
```

In this algorithm the Instrument Firmware function strengthens the firmware's capabilities by integrating defensive or monitoring methods by adding security-related features through instrumentation. It loads firmware as input. Then, using the instrumented firmware to run simulations and record behavioural data while it runs, the Collect Data function gathers and stores the data for later study. The Train Behavioural Profiles feature allows the system to identify patterns linked to both normal and abnormal behaviours by using past data to train machine learning models. The resulting models are essential to the algorithm's next stages. By integrating the firmware with the developed machine learning models, the Integrate with Firmware feature creates a more secure firmware version that has improved threat detection capabilities.

The Monitor Behaviour feature uses integrated machine learning models to continually monitor the secure firmware's behaviour and identify anomalies. When anomalies are found, an alarm is sent out. This sets off the Mitigate Anomalies mechanism, which examines and reduces every alarm in order to successfully handle possible security risks. Retraining machine learning models with fresh historical data is what the Update Behavioural Profiles function does to keep the system flexible in the face of changing security threats. Lastly, to provide continuous resistance against new attacks, the Run BBAD Algorithm function manages the continuous cycle of observing safe firmware behaviour and updating behavioural profiles. The Evaluate Effectiveness function, which comes at the end of the algorithm, evaluates the overall effectiveness of the security measures using risk analysis and actual case studies. This gives important information about how reliable the cybersecurity framework that has been put in place.

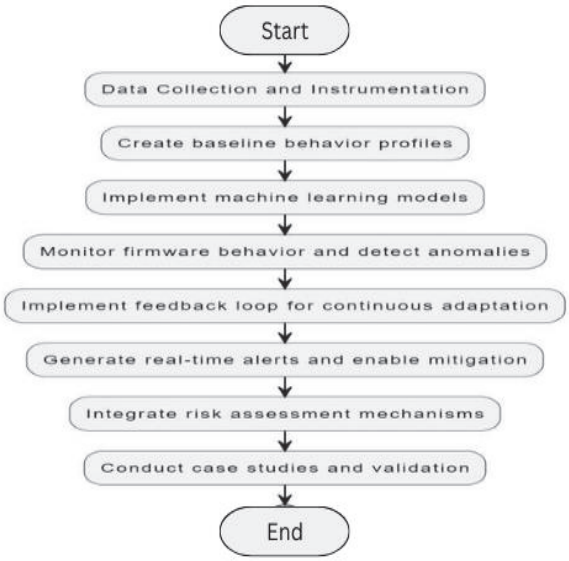

Fig. 2. Sequence Diagram of BBAD

B. Implementation

To implement BBAD, the first step is to instrument the target firmware to collect behavioral data at runtime. This requires adding monitoring and datacollection hooks to the firmware code.Next, machine learning models are trained using historical data to create baseline behavioral profiles for each firmware component. These models are then integrated into the security infrastructure of the firmware.

BBAD implementation steps are shown in Fig.3. During operation, BBAD continuously monitors the operation of the firmware components and compares their operation with the defined initial state. When deviations from normal behavior are detected, the system generates real-time alerts that enable immediate analysis and mitigation. Regular updates andrecalibrations of baseline behavioral profiles through a feedback loop ensure the adaptability and effectiveness of BBAD implementation over time.

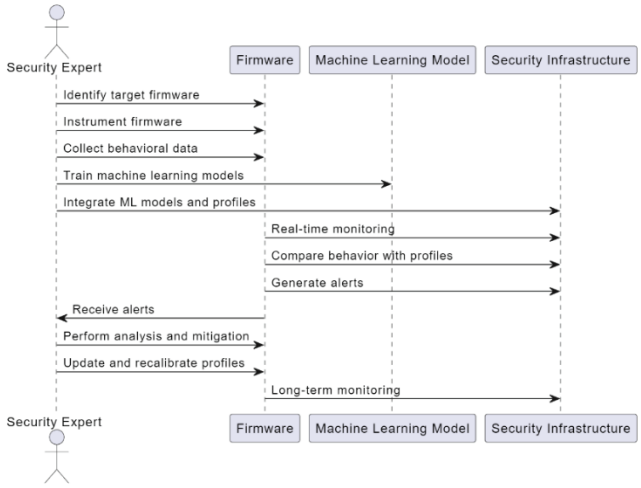

Fig. 3. BBAD implementation

The BBAD firmware security analysis method offers several key features that distinguish it as an innovative and effective approach. First, BBAD shifts the paradigm from traditional static and dynamic analysis to predictive behavioral detection, enabling real-time threat detection.

This is achieved by using advanced machine learning models for behavioral profiling, establishing baselines of normal behavior for firmware components, and continuous monitoring. The inclusion of a feedback loop ensures adaptability to evolving firmware, making it resilient to changing threats.

BBAD's investment in real-time alerts enables immediate mitigation actions that improve public safety. Importantly, this method combines risk analysis and supports its findingswith real case studies, demonstrating its effectiveness in patching firmware vulnerabilities. There are many cases of intruder trying to attack to access the data or breach the security.

C. Case Study - Smart Study Home:

Envision a scenario where a smart house security business installs a sophisticated smart door entry system using face recognition technology as a basis. In this instance, the business decides to use the Behavioural Based Anomaly Detection (BBAD) technique to strengthen firmware security. Experts in security work together to provide large datasets specifically designed for machine learning model training. These datasets provide a thorough basis for the creation and improvement of the BBAD methodology, including a wide range of situations such as authorised user behaviour, routine access patterns, and possible threat scenarios.

[i] Firmware Security Breach Attempt:

In a simulated attack scenario, an intruder attempted to manipulate the firmware by exploiting vulnerabilities in the facial recognition system. The attacker, posing as a stranger,aimed to gain unauthorized access by presenting a high- quality photograph during the facial recognition process.

[ii] BBAD Response:

BBAD, utilizing advanced ML models including anomaly detection algorithms and neural networks, responded effectively to the breach attempt. The security system, equipped with strategically placed sensors such as cameras, captured facial recognition data and environmental conditions.

The ML model was trained with liveness detection using the sensors, the behavioral data alreadycollected by the sensor checks the profile pattern, trained onhistorical data, detected anomalies in real-time during the facial recognition process.

[iii] Security Measures Implemented:

The security expert provided the API sensor with data that included diverse facial images, both normal and potential threat scenarios. The ML model, a convolutional neural network (CNN), learned to identify patterns associated with known attack vectors, setting a threshold for acceptable facial recognition patterns.

[iv] Identification of Potential Threat:

As the attacker presented a photograph, BBAD detected a deviation from the baseline behavioral profile. The neural network recognized the abnormality, triggering an immediate alert. The system, understanding the potential threat, denied access to the intruder, preventing unauthorized entry.

[v] Continuous Improvement:

BBAD's adaptability allowed it to constantly update and calibrate its birth biographies. This ensured that the system was flexible to evolving attack strategies. The real-time alert security team further analyzed the incident, contributing to the security of the firmware. The BBAD approach, which focuses on real-time monitoring, robust ML models and adaptive behavioral profiling, has proven to be highly effective in obfuscating implicit traps in a smart door entry system. By blocking access during a hardware security breach attempt, BBAD demonstrated its visionary position in strengthening smart home biases against evolving cyber security challenges.

D. Theoretical Analysis and Security Framework

For the [BBAD] implementation in a smart home security system as discussed in the case study , security experts collaborate to provide comprehensive datasets for machine learning model training. The behavioral model for this case is implemented based on TensorFlow.

[i] Data Set: A normal and several attacked firmware environments is simulated. The system behaviors are captured separately, and then used as the data set of the following experiment. [10] The types of attack mainly include IOT malware attack, where the intruder tries to attack the collected data or behavioral profiles, Bricker Bot [service denying attack], Photograph spoofing attack.

[ii] Functional Setup:

Functional steps involved are presented as follows:

1) Behavior Collection

At first, the normal behavioral patterns of the authorized users are to be simulated. To collect data, API sensors and monitoring devices such as camera and face recognition systems has to be employed to capture data on regular system behaviors, creating a baseline for normalcy. For potential malware attempting to manipulate or compromise the collected data or behavioral profiles, scenarios are created where the system is subjected to simulated IoT malware attacks. Data on intrusion attempts and potential manipulations are captured during these simulations. Additionally, a permanent denial-of-service attack using Bricker Bot is simulated to understand system behaviors under this specific threat.

In data collection, the system monitors and records the effects of the BrickerBot attack, documenting disruptions, failures, and responses. Simulated attacks are conducted where intruders attempt photograph spoofing to gain unauthorized access, involving the presentation of high-quality photographs during the facial recognition process. Instances where the system encounters photograph spoofing attempts are captured, and the system's response and any anomalies detected are recorded. TensorFlow is used to train the behavioral model based on collected data from both normal and attacked scenarios.

This process entails leveraging TensorFlow's capabilities for building and training neural networks[16]. The collected data from normal and attacked scenarios is utilized to train the TensorFlow-based behavioral model, ensuring it learns patterns associated with both normalcy and potential threats. A comprehensive dataset is compiled, incorporating data from normal behaviors, IoT malware attacks, BrickerBot attacks, and photograph spoofing attacks. The data collected from different scenarios is combined to create a diverse and representative dataset for training and evaluating the TensorFlow-based BBAD model.

2) Training Machine Learning models

The training of machine learning models within the BBAD framework involves utilizing the data collected from API sensors in a meticulous process. The API sensor feeds diverse and comprehensive collected datasets, encompassing normal and attacked firmware behaviors, into the training pipeline. The data undergoes pre processing, including feature extraction and transformation, optimizing it for model ingestion. TensorFlow's robust ecosystem is leveraged for implementing various neural network architectures. Convolutional neural networks (CNNs) are specifically employed for facial recognition model training, while recurrent neural networks (RNNs) [13] with Long Short-Term Memory (LSTM) networks handle IoT malware intrusion detection.

The most commonly used statistical analysis model for time series analysis and forecasting is the Auto Regressive Integrated Moving Average (ARIMA) [5]. It uses the time series data to gain an improved understanding of the dataset, or to anticipate future trends. This model helps to train the Bricker Bot Denial-of-Service detection module by capturing temporal patterns. Unsupervised learning techniques like k-means cluster help identify unusual user interaction patterns. Behavior profiling, which is essential for anomaly detection, can be achieved using Gaussian Mixture Models (GMMs)[11]. Iterative optimization helps train models by adjusting model parameters to reduce error and improve predictive accuracy. The trained models form the core of the BBAD, ready to detect anomalies and improve firmware security in real time.

3) Abnormal Behavior Detection

Under the BBAD paradigm, abnormal behavior detection is a complex process that relies on trained machine learning modelsthat analyze real-time data streams from API sensors. The facial recognition subsystem, powered by a CNN, analyzes facial feature vectors that are extracted from incoming images using techniques such as max pooling or non-linear activation to detect abnormal patterns [10]. The IoT malware detection module uses an RNN-LSTM architecture to analyze sequence data streams from API data streams to detect malicious intrusion attempts.The BrickerBot denies-of-service detection uses ARIMA models that analyze time series patterns to detect irregularities that indicate a potential attack. The k-means cluster is another layer of anomaly detection that looks for abnormal patterns in user interactions to improve the system's ability to identify subtle threats. In the final step, Gaussian mixture models (GMMs) are used for comprehensive behavioral profiling to further refine anomaly detection by setting normality baselines [9]. This multilayered approach provides the BBAD system witha nuanced understanding of threats and triggers real-time alerts to fortify firmware security.

IV. Comparative Analysis

In this case, the binary analysis approach may be useful for static vulnerability detection but lacks dynamic visibility. Dynamic analysis approach offers runtime visibility but is resource- intensive and may produce false positives. Fuzz testing approach is useful for input-related vulnerability detection but may miss more complex anomalies. BBAD shows superiority for capturing dynamic and changing threats, so it is a proactive option for improving firmware security. In Fig 4, the comparative analysis is provided with hypothetical scores for their approach in the above case study mentioned in section III.

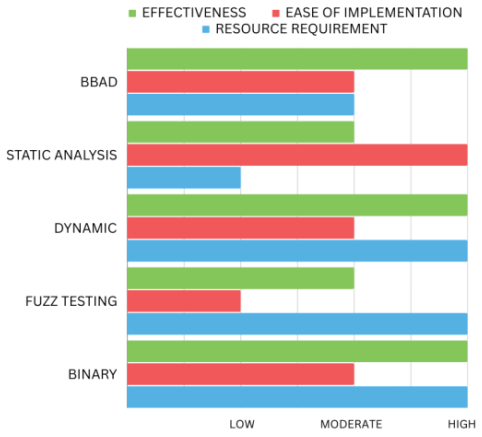

Fig. 4. Comparative analysis of various approaches

V. Conclusion

In conclusion, BBAD provides a comprehensive and effective approach that will revolutionize firmware security assessment and make it a major advance in the security of modern computing devices. BBAD offers several distinct advantages over existing firmware security analysis methods. First, it works in real time, reducing the need for resource-intensive static and dynamic analyses, thus increasing efficiency. Second, the proactive nature of BBAD allows you to detect threats in their early stages and proactively respond to potential vulnerabilities. Additionally, its adaptability to firmware updates and evolving threats through a feedback loop ensures long-term effectiveness. BBAD prioritizes detecting anomalies in firmware behavior over known vulnerabilities, making it resilient against zero-day attacks and emerging threats. Finally, BBAD's focus on continuous behavioral monitoring and real- time alerts improves the overall security of modern computing devices.

VI. Future Work And Considerations

Future developments in Behavioral Based Anomaly Detection (BBAD) approach may involve the integration of additional sensory data, exploration of ensemble learning techniques to improve detection accuracy, resolution of scalability concerns, and development of user-friendly interfaces for security administrators. Future research will focus on improving security to counteract all potential threats, including supply chain attacks, radio frequency (RF) attacks, side channel attacks, and other attacks that are integrated into firmware security. Additionally, investigating ensemble learning strategies might be done to integrate the advantages of different machine learning models within BBAD. These methods aim to increase BBAD's adaptability and utility in dealing with evolving cybersecurity problems in the future.

References

[1] S. Hemram, G. J. W. Kathrine, G. M. Palmer and S. E. V. Ewards, "Firmware Vulnerability Detection in Embedded Systems and Internet of Things," 2022 International Conference on Augmented Intelligence and Sustainable Systems (ICAISS), Trichy, India, 2022, pp. 1161-1167, doi: 10.1109/ICAISS55157.2022.10010804.

[2] N. S. Mtetwa, P. Tarwireyi, A. M. Abu-Mahfouz and M. O. Adigun, "Secure Firmware Updates in the Internet of Things: A survey," 2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC), Vanderbijlpark, South Africa, 2019, pp. 1-7, doi:10.1109/IMITEC45504.2019.9015845.

[3] M. Sarikonda and S. R, "Validation of Firmware Security using Fuzzing and Penetration Methodologies," 2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon), Vijaypur, India, 2022, pp. 1-5, doi: 10.1109/NKCon56289.2022.10126524.

[4] A. Kolehmainen, "Secure Firmware Updates for IoT: A Survey," 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Halifax, NS, Canada, 2018, pp. 112-117, doi: 10.1109/Cybermatics_2018.2018.00051.

[5] Kadri, Mohamed Riadh & Abdelli, A. & othman, Jalel & Mokdad, Lynda. (2023). Survey and classification of Dos and DDos attack detection and validation approaches for IoT environments. Internet of Things. 101021. 10.1016/j.iot.2023.101021.

[6] J. Menon, C. Hauser, Y. Shoshitaishvili and S. Schwab, "A Binary Analysis Approach to Retrofit Security in Input Parsing Routines," 2018 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 2018, pp. 306-322, doi:10.1109/SPW.2018.00049.

[7] J. Liang, M. Wang, Y. Chen, Y. Jiang and R. Zhang, "Fuzz testing in practice: Obstacles and solutions," 2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER), Campobasso, Italy, 2018, pp. 562-566, doi: 10.1109/SANER.2018.8330260..

[8] Mingzhe Wang et al., "SAFL: Increasing and Accelerating Testing Coverage with Symbolic Execution and Guided Fuzzing", Software Engineering Companion (ICSE-C) 2018 IEEE/ACM 40th International Conference on Software Engineering, 2018.

[9] M. Singh, B. Mehtre and S. Sangeetha, "User Behaviour based Insider Threat Detection in Critical Infrastructures," 2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC), Jalandhar, India, 2021, pp. 489-494, doi: 10.1109/ICSCCC51823.2021.94781

[10] Meng, W., Wang, J., Liu, C., Xu, J., Wang, J., Hao, S., Yi, W., Zhong, J. (2022). IoT-DeepSense: Behavioral Security Detection of IoT Devices Based on Firmware Virtualization and Deep Learning. Security and Communication Networks, 2022, 1443978.

[11] V. Kilaru, M. G. Amin, F. Ahmad, P. Sévigny and D. DiFilippo, "Gaussian mixture model based features for stationary human identification in urban radar imagery," 2014 IEEE Radar Conference, Cincinnati, OH, USA, 2014, pp. 0426-0430, doi: 10.1109/RADAR.2014.6875628.

[12] Nadir, Ibrahim & Mahmood, Haroon & Shah, Ghalib. (2022). A taxonomy of IoT firmware security and principal firmware analysis techniques. International Journal of Critical Infrastructure Protection. 38. 100552. 10.1016/j.ijcip.2022.100552.

[13] Zaremba, Wojciech & Sutskever, Ilya & Vinyals, Oriol. (2014). Recurrent Neural Network Regularization.

[14] Greco, Claudia & Ianni, Michele & Guzzo, Antonella & Fortino, Giancarlo. (2023). Firmware Dynamic Analysis Through Rewriting. 10.1007/978-3-031-42194-5_2.

[15] Pistoia, Marco & Chandra, S. & Fink, Stephen & Yahav, Eran. (2007). A survey of static analysis methods for identifying security vulnerabilities in software systems. IBM Systems Journal. 46. 265-288. 10.1147/sj

[16] H. Ben Braiek and F. Khomh, "TFCheck : A TensorFlow Library for Detecting Training Issues in Neural Network Programs," 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS), Sofia, Bulgaria, 2019, pp. 426-433, doi: 10.1109/QRS.2019.00059.