Lightweight Node-level Malware Detection and Network-level Malware Confinement in IoT Networks

Sai Manoj Pudukotai Dinakarrao[1], Hossein Sayadi[1], Hosein Mohammadi Makrani[1],
Cameron Nowzari[2], Setareh Rafatirad[2] and Houman Homayoun[1]
[1]Department of Electrical and Computer Engineering
[2]Department of Information Sciences and Technology
George Mason University, Fairfax VA, USA 22030

***Abstract*—The sheer size of IoT networks being deployed today presents an "attack surface" and poses significant security risks at a scale never before encountered. In other words, a single device/node in a network that becomes infected with malware has the potential to spread malware across the network, eventually ceasing the network functionality. Simply detecting and quarantining the malware in IoT networks does not guarantee to prevent malware propagation. On the other hand, use of traditional control theory for malware confinement is not effective, as most of the existing works do not consider real-time malware control strategies that can be implemented using uncertain infection information of the nodes in the network or have the containment problem decoupled from network performance. In this work, we propose a two-pronged approach, where a runtime malware detector (*HaRM*) that employs Hardware Performance Counter (HPC) values to detect the malware and benign applications is devised. This information is fed during runtime to a stochastic model predictive controller to confine the malware propagation without hampering the network performance. With the proposed solution, a runtime malware detection accuracy of 92.21% with a runtime of 10ns is achieved, which is an order of magnitude faster than existing malware detection solutions. Synthesizing this output with the model predictive containment strategy lead to achieving an average network throughput of nearly 200% of that of IoT networks without any embedded defense.**

***Index Terms*—Malware detection, IoT networks, Malware confinement, Network security**

I. Introduction

Amelioration of miniature computing devices into the consumer and industrial markets with enabled connectivity to the Internet towards smart and intelligent features lead to an upsurge in the size of networks through which they are linked and communicate [1]. It is predicted that by the year 2020, there will be nearly 50 billion devices connected to the Internet-of-Things (IoT) [2]. Different features including the communication using heterogeneous protocols, low-power, miniature footprint, and the mobility features offered by many of these IoT devices make them an attractive choice for numerous applications ranging from fitness tracking and health monitoring, to defense and security purposes. Along with the legion of benefits offered by IoT devices comes an equal amount of potential vulnerabilities and security risks that have never been experienced [3], [4].

Along with the primary functionality of communication, IoT devices also share underpinning security risks of the Cyber-Physical Systems (CPS), but are more vulnerable for such attacks as the security is often neglected in IoT device design. Consequently, as the IoT devices are online with very minimal protection measures [5], even if any, exposes them to potential cyber-attacks. This makes them vulnerable for security attacks, and it is not feasible to deploy existing software-based malware detection due to the resource limitation. Furthermore, from the adversaries' perspective, the feasibility for malware propagation via connected network with none/weakly built defense measures, and a vast connectivity makes the IoT devices a potential target for attacks [3], [4]. These attacks can be targeted at various devices such as routers and CCTVs.

Another potential security risk in IoT networks is the feasibility for malware to spread across the network as soon as one of the devices/nodes is infected, resulting in the whole network being compromised. Unfortunately, given the multitude of IoT networks that are being deployed, it is impractical to quarantine the malware in the infected systems as they would have propagated to other devices already. More specifically, the accentuating size and popularity of these networks further exacerbates the challenge of securing IoT devices and restricting the malware propagation, as we no longer have the option to just 'restart' the entire system as it is too large and the cost of restarting a massive network may easily exceed the cost of potential malware existing in the network. This propagation of malware through the IoT network can hobble the network performance such as throughput.

Coalescing the above discussed malware detection on an IoT node, and propagation of malware in the network reveals significant issues that have to be addressed before large-scale global deployments of IoT networks can be realized. To address this, we propose a unified solution that addresses both the challenges: a) develop and deploy lightweight malware detection on IoT devices without incurring large overheads, and b) confine the propagation of malware in the IoT network even with the imperfect infection information while preserving network integrity and overall performance.

For malware detection in IoT nodes, we deploy a hardware performance counter (HPC) based runtime malware detector (*HaRM*) that utilizes lightweight machine learning (ML) classifiers for detecting malware. To confine the malware propagation through the network during its detection, we utilize a low complex heuristics based approach to determine the connectivity between nodes to minimize the malware spreading without hobbling the overall network throughput. In contrast to employing the speculative information of malware propagation, we combine the malware propagation model information with imperfect information from *HaRM*for confining the malware in the network.

Contributions: The main contributions of this work can be outlined in a three-fold manner as follows:

- The proposed *HaRM*employs low computational overhead ML classifier (OneR) to circumvent high resource utilization and suit the needs of IoT devices, also achieves good malware detection accuracy.
- We develop a framework for combining the models of malware spreading processes on networks explicitly with their direct adverse effects on network performance and formulate a optimal control problem for malware confinement while maintaining network integrity.
- We show how the output of the *HaRM*detector can be used to generate imperfect estimates of infection state information as an input to control the malware spread.

II. Network Description and Problem Formulation

Here, we introduce the malware propagation model, and the malware confinement problem. The overall network architecture with node-level malware detector and the network-level malware epidemic controller is depicted in Figure 1.

A. Modeling Malware

To emulate the real-world malware spreading where the malware performs the devised activity such as unauthorized data transfer and gets deactivated [6] , we use the Susceptible-Infected-Susceptible (SIS) model. Some of the real-world malware that follows this kind of model are 'badBios' [7], 'Yankee Doodle' [8], and 'Magneto' [9]. Susceptible-Infected-Susceptible (SIS) model is a well-established model for epidemics on networks [10], [11]. Here, the network is represented as a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$ with $|\mathcal{V}| = n$ nodes, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ as the directed edges, and $W \in \mathbb{R}^{n \times n}$ is the weighted adjacency matrix. The edge $(i, j) \in \mathcal{E}$ means that node j is sending data to node i at a rate proportional to w_{ij}. Note that $w_{ij} > 0$ if and only if $(i, j) \in \mathcal{E}$, and $w_{ij} = 0$ otherwise. We denote the set of neighbors of i as $\mathcal{N}_i = \{j \in \mathcal{V} | (i, j) \in \mathcal{E}\}$. At any given time, the set of nodes are split into two *compartments*: Susceptible (S) and Infected (I), that represents the infection state of each node. The state of node i at time t is given by the binary random variable $X_i(t) \in \{0, 1\}$, where $X_i(t) = 1$ indicates that the node i is infected with malware at time t, and similarly, $X_i(t) = 0$ indicates that the node i is currently free of malware, but susceptible. The infection state of the entire network is denoted by a vector $X(t) \in \{0, 1\}^n$.

The intuition of malware spreading model is as follows. Any node i that is infected with the malware is capable of passing it to a neighbor $j \in \mathcal{N}_i$ randomly with Poisson rate $\beta > 0$ proportional to the amount of traffic flow w_{ij}, β termed as the infection rate. At the same time each infected node is also able to recover with Poisson rate $\delta > 0$, δ termed as the recovery rate. Thus, the SIS spreading process can be modeled using the Markov process as follows

$$\begin{aligned} X_i &: 0 \to 1 \quad \text{with rate } \beta \textstyle\sum_{j \in \mathcal{N}_i} w_{ij} X_j, \\ X_i &: 1 \to 0 \quad \text{with rate } \delta. \end{aligned} \tag{1}$$

B. Malware Epidemic Control Problem

Based on the introduced models, we formulate the problem of malware epidemic control in the network here. Let $W'(t)$ denote the modified graph where the traffic between some nodes may have been reduced due to different control activities such as removal of links. In other words, as a consequence of the control mechanism the traffic is regulated between each pair of active links $w'_{ij} \leq w_{ij}$ to reduce the chance of node j spreading malware to node i. Thus, the problem can be defined as maximizing the objective function

$$J = \frac{1}{T} \int_0^T P(W'(t), X(t)) dt \tag{2}$$

over some time horizon $T > 0$. Here $P(W'(t), X(t))$ denotes the performance of the network at time-instant t. Note that this objective function explicitly captures the trade-off between shutting down links to contain the malware at the cost of reducing instantaneous network performance, and keeping links active to maintain the instantaneous network performance at the risk of letting the malware spread.

This problem poses two challenges. First, in general, we may not have access to the true infection state $X(t)$ of a node, i.e. malware is often meant to be undetected. This indicates that a mechanism to detect the malware on nodes is needed. Secondly, the malware confinement can reduce the malware spread through the network, but can impact the performance. In other words, malware confinement might not hamper instantaneous network performance, but can impact after certain time. Similarly, shutting down the links can confine malware propagation, but impacts the instantaneous network performance. Furthermore, the whole problem of malware confinement involves challenge of considering the true state of the nodes, infection spread characteristics of malware in the network, and a way to combine them.

Optimizing equation (2) is non-trivial even if the infection state $X(t)$ of the network is known at all times. Thus, to solve the problem of malware containment in the network, we partition it into two subproblems: (a) detect the malware on the nodes without incurring large overheads; and (b) confine the malware based on the output of malware detection algorithm.

III. Node-level Runtime Malware Detection

To perform effective node-level malware detection in IoT network, we propose a lightweight, hardware-assisted runtime malware detection technique (*HaRM*).

A. Overview of *HaRM*

The proposed malware detector, *HaRM* is shown in Figure 2. It comprises of feature collection and selection (performed offline), and runtime malware detection (performed online).

*1) **Feature Selection**:* For runtime malware detection, we employ HPC traces in this work, as software-based malware detection incurs latency and processing overheads [12], [13], [14]. As discussed, there exists a large number of microarchitectural events, that can be monitored. However, given the limited number of available HPCs in today's microprocessors, at a max 4 for most IoT devices, only few microarchitectural events equal to the number of available HPCs can be monitored simultaneously. In addition, using all microarchitectural events incur large computational overheads. Therefore, a subset of HPCs are selected that represent the most critical features required for malware detection. This feature selection process is performed offline.

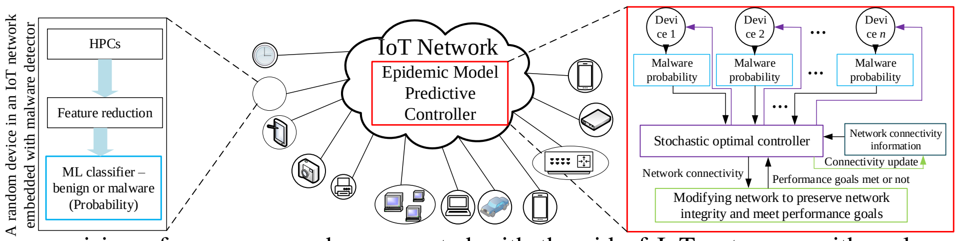
A random device in an IoT network embedded with malware detector
HPCs
Feature reduction
ML classifier – benign or malware (Probability)
IoT Network
Epidemic Model Predictive Controller
Device 1
Device 2
Device n
Malware probability
Malware probability
Malware probability
Stochastic optimal controller
Network connectivity information
Connectivity update
Network connectivity
Performance goals met or not
Modifying network to preserve network integrity and meet performance goals

Fig. 1.: IoT network comprising of numerous nodes connected with the aid of IoT gateways with malware detector deployed on each node and a centralized malware confiner

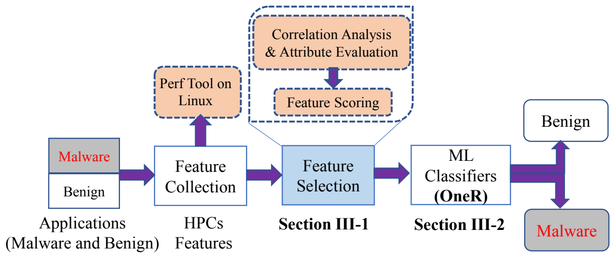

Fig. 2.: Overview of the proposed node-level *HaRM*

We collected 44 possible diverse microarchitectural events using the available HPCs for the employed IoT devices by repeating the experiments multiple times. For feature reduction, we apply "Correlation Attribute Evaluation" to identify the most critical HPC events. Correlation evaluation algorithm calculates the Pearson correlation coefficient between each attribute and class, as given below.

$$\rho(i) = \frac{cov(Z_i, C)}{\sqrt{var(Z_i)\ var(C)}} \qquad i = 1, ..., 44 \tag{3}$$

where ρ is the Pearson correlation coefficient. Z_i is the input dataset of event i ($i = 1, \dots, 44$). C is the output dataset on each IoT node containing labels, i.e. "Malware" or "Benign" in our case. The $cov(Z_i, C)$ measures the covariance between input and output data. The $var(Z_i)$ and $var(C)$ measure variance of both input and output datasets, respectively.

Based on the ranking of ρ, top 8 features are selected for detecting of malicious applications on the IoT device. The features are ranked based on their importance and relevance to the target variable through the feature scoring process. The reduced set of features indicate that the top 8 includes HPCs events representing pipeline front-end, pipeline back-end, cache subsystem, and main memory behaviors which are influential in the performance of standard applications. The 'Branch instruction' being the most influential event to classify malware from benign applications. Further, as 16 HPCs are not affordable for IoT devices, Principal Component Analysis (PCA) is applied on the 16 features to select the prominent 4.

2) ***Malware Detection using ML Classifiers:*** Once the key features are selected, they are used to train the ML classifiers i.e. malware detectors of *HaRM*. For evaluation, we experimented with various ML classifiers and compare them in terms of malware detection accuracy, hardware overhead, power consumption, and the time required to detect malware (latency). The running application is profiled every 1ms i.e. non-trivial HPCs are collected continuously at 1ms interval and fed to the ML classifier. A k-fold (k=10) validation is employed for evaluating and comparing the malware detection accuracy of different classifiers. Training the malware detectors involve profiling the incoming application with *Perf* tool available under embedded Linux and extracting the vital performance counters that are determined by the feature selection (as in Section A1), and deriving a learning model from the training data. The output variable is the class (malware or benign) of an application (as shown in Figure 2).

Given the derived detection model, the ML classifiers provide information regarding the existence of malware. For the IoT devices, we employ 'OneR' ML classifier for node-level malware detection due to its lower latency and resource consumption (as shown in Section B). As the malware detection is performed on individual nodes, it is independent of the network topology. As seen in the evaluation, the OneR based *HaRM* achieves only 92% accuracy, indicating that there is feasibility to have false positive or negatives. However, as the deployed epidemic controller for malware confinement requires an estimate of infection rather than deterministic infection state of the node, using of *HaRM* is still beneficial.

IV. Malware Epidemic Control

Despite the best efforts in malware detection including the deployed *HaRM*, there exist no technique that has a perfect yield, as such it is not always possible to have the exact infection data $X_i(t)$ for node $i \in \mathcal{V}$.

First of all, the malware propagation model (SIS) only provides independent estimates $\hat{X}_i(t) = \Pr(X_i(t) = 1)$. Since the true infection state $X_i(t)$ is a binary random variable for each node i, we instead maintain an estimate $\hat{X}_i(t) \in [0, 1]$ at all times. More specifically, we let $\hat{X}_i(t|t)$ be the estimate of infection state $X_i(t)$ available at time t.

Taking expectations of the random binary infection variables, the dynamics according to (1) are given by

$$\frac{d\mathbb{E}[X_i](t)}{dt} = -\delta X_i(t) + \beta \sum_{j \in \mathcal{N}_i} w'_{ij} X_j(t) \tag{4}$$

where the modified network W' is used rather than the original network W. In addition, we need to combine this estimate with the updated information provided by the malware detectors (*HaRM*) embedded on the IoT nodes. Using this, we can propagate true estimates forward in time.

The infection states due to the malware on applications executed by the devices have to be considered to capture the true state of the nodes. As aforementioned, *HaRM*does not provide perfect information, rather an estimate is provided. These estimates (outputs of the *HaRM*) have to be combined with the malware propagation model estimate in order to obtain independent estimates $\hat{X}_i(t|t) = \Pr(X_i(t) = 1)$ each time the malware detection algorithm is run.

Let $y_i(t_\ell) \in [0, 1]$ be the output of the malware detector (***HaRM***) on node i at time t_ℓ. Assuming this output is an independent probability that the node is infected with malware, we update the probability of infection of each node $\hat{X}_i(t_\ell|t_\ell)$ conditioned on the new information available at each sampling time $t \in \{t_\ell\}_{\ell \in \mathbb{Z}_{\geq 0}}$ as

$$\hat{X}_i(t_\ell|t_\ell) = 1 - ((1 - \hat{X}_i(t_\ell|t_{\ell-1}))(1 - y_i(t_\ell))) \tag{5}$$

Thus, given both a way for propagating the estimate $\hat{X}(t'|t)$ at time t and a way to incorporate new measurements $y_i(t_\ell)$, now we estimate $\hat{X}(t)$ of the infection state of all nodes.

Given the current infection estimate $\hat{X}_i(t_\ell|t_\ell)$, we propagate this according to equation (4) at a smaller time later by

$$\begin{aligned}\hat{X}_i(t_{\ell+1}|t_\ell) = \hat{X}_i(t_\ell|t_\ell) + \Delta t(-\delta \hat{X}_i(t_\ell|t_\ell) + \\ \beta \sum_{j \in \mathcal{N}_i} w'_{ij} \hat{X}_j(t_\ell|t_\ell))\end{aligned} \tag{6}$$

Thus, the objective of optimal control problem in equation (2) is redefined as a rate-constrained problem as follows.

Theorem 4.1: [Equivalent problem]

$$\begin{aligned} \underset{\{w'_{ij}\}_{(i,j)\in\mathcal{E}}}{\text{minimize}} \quad & \sum_{(i,j)\in\mathcal{E}} g_{ij}(w'_{ij}) \\ \text{s.t.} \quad & \sum_{i\in\mathcal{O}} (1-\delta) X_i + \psi_i^{\{w\}}(1 - X_i) \leq r \sum_{i\in\mathcal{V}} \hat{X}_i(t|t), \end{aligned} \tag{7}$$

where we have defined the convex function

$$\psi_i^{\{w\}} = 1 - \hat{X}_{j'}(t|t)\gamma_{j'i}^{\frac{1}{w}} \Pi_{j\in\{\mathcal{N}_i \cap \mathcal{X}_i\}}\, \gamma_{ji}^{\frac{1}{w}} - \hat{x}_{j'}^{c}(t|t) \Pi_{j\in\{\mathcal{N}_i \cap \mathcal{X}_i\}} \gamma_{ji}^{\frac{1}{w}} \tag{8}$$

where $w > d_{\max}$ and $\mathcal{X}_i = \{i \in V \cap \mathcal{O} \,|\, X_i = 1\}$. Suppose the function g_{ij} is convex in the variable w'_{ij}, (7) is the equivalent optimization problem for defined problem statement, where the optimal edge weights can be computed as $w'_{ij} = 1 - (w^\star_{ij})^{\frac{1}{w}}$, where $w^\star_{ij}$ is the solution to (7).

However, it is worth noting that product terms are in general non-convex, and so CVX [15] may not solve the problem.

Finally, to put the solutions to the 2 subproblems together in a way to solve the optimization problem (2), we need to consider a specific form of the performance function P. There exists numerous metrics to evaluate the performance of a network such as throughput, latency, and bandwidth. In this work, to evaluate the network performance in terms of overall network throughput considering the infection state of nodes, given by [16]

$$P(W'(t), X(t)) = \tau = \int_0^T \frac{\log(1 + w)}{h} P_{suc}(1 - X(t)) \tag{9}$$

where P_{suc} is the probability of successful transmission (i.e. malware free data), modeled as the inverse of the node malware infection probability; h is the number of hops; and w is the weight assigned to the communicating nodes (signal-to-noise ratio (SNR)). The SNR is given by

$$\frac{g_{ij} d_{ij}^{-\alpha}}{\sum_{k\in L(t)\backslash j} \; g_{ij} d_{ik}^{-\alpha}} \tag{10}$$

where d_{ij} represents the distance between nodes i, j with a channel gain g_{ij} in the network $L(t)$ at a given time t; The path-loss exponent given by α. As such, the traffic between nodes is determined based on the throughput, connectivity, and the malware infection of the nodes.

Depending on the determined malware infection state and the infection spread, the infected node with higher number of connected neighbors are chosen. The links from those nodes are regulated i.e. traffic from infected node j to neighbors, say i are made to be $w_{ij} \times r$. The performance constraint is imposed when regulating the traffic neighbors of the infected node i.e. if the performance drops below the required performance, the traffic regulation process is ceased. The decay rate r is provided to the epidemic controller based on the malware threat parameter σ as a way to limit how much the network can be disconnected. As $r \to 0$, the solution to Theorem 4.1 provides much more aggressive containment strategies by disconnecting the entire network. On the other hand, as $r \to 1$, the solution to Theorem 4.1 allows the malware to spread and not disconnect many links. Based on this, the proposed solution determines the best possible network connectivity in order to maintain as many original connections as possible while satisfying the desired decay rate.

V. Experimental Results

A. Experimental Setup

The experimental setup for deploying the IoT network is discussed here. The malware propagation performed in the experiments is similar to that in [17], [18].

Network and Hardware Setup: 20 different IoT nodes such as temperature sensors connected with Intel ATLASEDGE board, Beagle Boards (BeagleBone Blue) having ARM processor, communicating via Bluetooth protocol are deployed in an area of 5×5 m^2. Most of the deployed boards host RISC architecture with embedded Linux OS running on them. The devices are statically placed during the experiments. The epidemic model predictive controller is executed on a controller built on Intel Haswell core i5 processor.

Software Framework: On each device, i.e. at the node-level, in order to extract the HPC information, *Perf* tool available under Linux is employed. Similar applications that are deployed on IoT nodes are executed on a system running Ubuntu 14.04 with Linux 4.4 Kernel. HPC information is obtained by executing all the applications in Linux Container (LXC) which is an isolated environment. LXC is chosen over other commonly available virtual platforms such as VMWare or VirtualBox since it provides access to actual performance counters data instead of emulating HPCs. We extracted 44 CPU events available under *Perf* tool. As the employed processors on IoT devices have only 4 HPC registers available [19], we can only measure 4 events at a time and thus, we run applications multiple times to capture all the events.

Applications: We executed 3000 benign and malware applications for HPC data collection. Benign applications include MiBench [20] and SPEC2006 [21], Linux system programs, browsers, text editors, and word processor. For malware applications, Linux malware is collected from virustotal.com

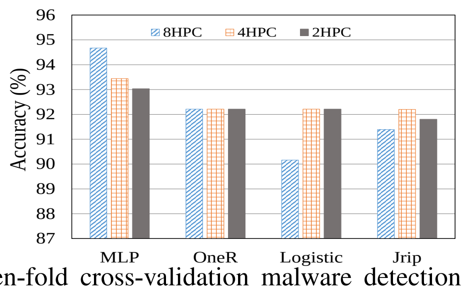

Fig. 3.: Ten-fold cross-validation malware detection accuracy with different ML classifiers and HPCs in the proposed *HaRM*

and classified on virusshare.com. Malware applications include Linux ELFs, python scripts, perl scripts, and bash scripts, which are created to perform malicious activities consisting of four classes of malware including 452 Backdoor, 350 Rootkit, 650 Virus, and 1169 Trojans.

B. Evaluation of HaRM: Node-Level Malware Detection

*1) **Malware Detection Accuracy**:* We evaluate the malware detection accuracy when different ML classifiers are employed in *HaRM* framework shown in Figure 2. A 10-fold cross-validation is used to verify the performance of malware detection with reduced features i.e. *HaRM*. For the detection accuracy, we calculate the percentage value of samples that are correctly classified. Figure 3 shows a comprehensive (average) accuracy comparison of various ML classifiers used for malware detection, with varied number of HPCs and ML classifiers. The *HaRM* achieves nearly 94.7% detection accuracy (max) when 8 HPCs are employed, and 93.03% when 2 HPCs are employed with the verified classifiers. As the IoT devices are constrained on the resources, using 8 HPCs is not feasible. Heavy weighted Multi-Layer Perceptron (MLP) classifier outperforms other classifiers even when less number of HPCs. Lightweight OneR classifier provides sufficient ($\sim$92.21%) and nearly constant accuracy despite varying number of HPCs. This is because, OneR employs only one prominent feature for classification, despite multiple HPCs are provided as inputs. We anticipate that the Logistic regression and JRip are underperforming with 8 HPCs due to the involved complex relationship between HPCs which require large amount of data points (malware runtime is generally very short, so limited data points can be captured).

*2) **Processing Overheads**:* Though a majority of the deployed classifiers have shown a sufficient malware detection accuracy, the resource consumption and the latency involved for malware detection play a crucial role in choosing the best suited ML model for runtime malware detection in *HaRM*. The *HaRM*'s hardware footprint is evaluated on a FPGA for a fair comparison. We use Vivado HLS to develop the HDL implementation of the classifiers (*HaRM*) and deploy on Xilinx Virtex 7 FPGA.

Figure 4 shows the malware detection power consumption and latency for different ML models when used in *HaRM*. One can observe that MLP has the highest power consumption and latency, whereas OneR has the least power consumption and smaller latency. The latency here refers to the processing time to capture a malware. In terms of power, the OneR has only 25% of power consumption compared to MLP, and 30% lower

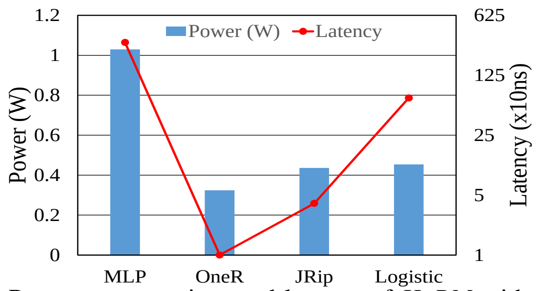

Fig. 4.: Power consumption, and latency of *HaRM* with different ML classifiers

compared to JRip or Logistic regression implementations. Furthermore, the malware detection accuracy degradation is $<2\%$ with 2 or 4 HPCs using OneR compared to the highest possible accuracy with MLPs, which is affordable in the current work, as the employed malware confinement requires an estimate of the infection state of the node rather than a high accurate deterministic value to perform the stochastic model predictive control. Also, it has been observed that OneR employs only one HPC value (feature) i.e. branch instructions for classification, which is possible to capture even in most of the low-end IoT devices. Branch instructions are one of the non-trivial microarchitectural events as most of the malware [22], [23] relies on branching operations for executing the malicious activity and the branch instructions also reveal the behavior of most of the malware. As such, based on the discussed analysis we choose to deploy OneR as the classifier for *HaRM* on the IoT devices for further evaluation.

C. Evaluation of Malware Epidemic Control

We evaluate the performance of malware epidemic controller on the network with 20 nodes deployed in $5 \times 5\text{m}^2$ area. Nearly 1000 experiments are carried out with each experiment lasting for $\sim$40 seconds. Any device in the network is affected by malware randomly at any point of time, and multiple attacks are deployed on each device to replicate real-world scenarios. At the initial time-instant ($t = 0$), two of the nodes are deployed with self-propagating malware. We evaluate the network performance in terms of overall average throughput.

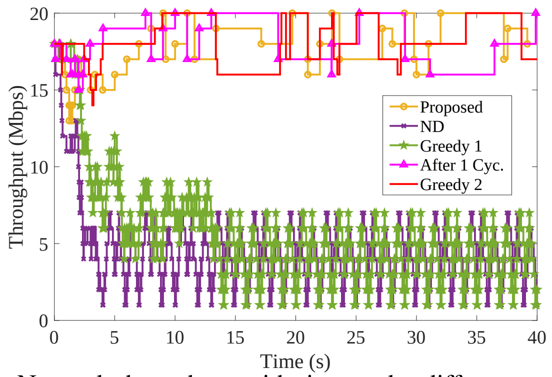

Fig. 5.: Network throughput with time under different malware confinement techniques

Figure 5 presents the network performance (throughput) of the proposed epidemic predictive controller i.e., network-level malware confinement provided with real-time malware infec-

TABLE I: Network throughput under different malware confinement schemes

Technique	Proposed	After 1 cycle	No disc.	Greedy 1	Greedy 2
Thr. (Mbps)	460.9631	418.9087	182.143	310.6328	243.2209

TABLE II: Average number of infected nodes in the network

Technique	Proposed	After 1 cycle	Greedy 1	Greedy 2
#	2.48	3.32	14.79	2.71

tion, state of the nodes in the network, and the performance when other heuristic methods are deployed in the IoT network as a defense for malware propagation. The deployed heuristics are: disconnect the node after 1 cycle of malware propagation (denoted as 'After 1 cyc.'); no defense in the network ('ND'); greedy algorithms based on the malware infection probability ('Greedy 1') i.e., disconnect the node if malware is detected by *HaRM* with a probability higher than a threshold (0.75); and based on the degree of infected node ('Greedy 2') i.e., disconnect infected node with highest neighbors. One can observe that with the proposed technique, the throughput stays close to the maximum bound. The greedy 1 and no disconnect performs worse, as the malware propagates through the network. The greedy 2 i.e. disconnecting the infected node with highest neighbors performs better than Greedy 1 indicating that infection spreads much faster than quarantining the malware and malware propagation has more impact on the overall network throughput. The network throughput obtained for the experiments deployed with various schemes is listed in Table I with first row describing the scheme for malware confinement, and the second row provides the overall network throughput (Thr in Mbps) for 40s. Nearly 200% throughput is achieved compared to network without having any defense for malware propagation defense. Similarly, up to 100% is achieved with proposed malware epidemic control provided with real-time infection data compared to heuristic approaches. Additionally, the amount of infected nodes after 40s are averaged for all the conducted experiments and presented in Table II. One can see that the number of infected nodes with proposed control mechanism has the least infected number of nodes after 40s.

VI. Conclusion

Despite the tremendous growth in IoT networks, security is often neglected in the design of IoT devices, making them vulnerable for cyber-attacks. A single compromised node in an IoT network can infect other nodes in the network, as a consequence of malware spread. We propose a novel practical solution for securing IoT networks against malware epidemics. In this work, a lightweight runtime malware detector referred as (*HaRM*), is deployed on IoT nodes for detecting malware. The proposed *HaRM* utilizes on-chip HPCs' information to detect malware without incurring processing overheads while facilitating runtime malware detection. Unfortunately, since the malware detection algorithms are not perfect, their outputs cannot be immediately used in theoretical optimal control problems. Instead, we use the outputs of *HaRM* to generate probabilistic outputs about the infection state rather than binary deterministic ones that can be used to determine the node disconnect criteria The proposed method is experimented in a 20 node IoT network deployed in a $5{\times}5\text{m}^2$ area. The deployed *HaRM* with OneR classifier achieves a malware detection accuracy of $\sim$92% on average, with a malware detection latency of 10ns. The proposed malware epidemic control method achieves a throughput of nearly $2\times$ compared to IoT network without any defense, and up to $1\times$ higher compared to other heuristic approaches.

References

[1] A. K. Sikder *et al.*, "A survey on sensor-based threats to internet-of-things (IoT) devices and applications," *CoRR*, vol. abs/1802.02041, 2018.

[2] D. Evans, "The internet of things: How the next evolution of the internet is changing everything," *CISCO White Paper*, pp. 1–11, Apr 2011.

[3] T. Abera *et al.*, "Things, trouble, trust: On building trust in iot systems," in *ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2016.

[4] J. Wurm *et al.*, "Security analysis on consumer and industrial IoT devices," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2016.

[5] M. B. Barcena and C. Wueest, "Insecurity in the internet of things," *Whitepaper*, Apr 2015.

[6] D. Chakrabarti *et al.*, "Epidemic thresholds in real networks," *ACM Trans. Inf. Syst. Secur.*, vol. 10, no. 4, pp. 1:1–1:26, Jan 2008.

[7] (2013) badbios. Last accessed: 29-Nov-2018. [Online]. Available: https://arstechnica.com/information-technology/2013/10/meet-badbios-the-mysterious-mac-and-pc-malware-that-jumps-airgaps/

[8] (2007) Yankee doodle. Last accessed: 29-Nov-2018. [Online]. Available: https://www.symantec.com/security-center/writeup/2000-121914-2303-99

[9] (2017) Magneto. Last accessed: 07-Aug-2018. [Online]. Available: https://magento.com/security/tag/malware

[10] R. J. Kryscio and C. Lefévre, "On the extinction of the SIS stochastic logistic epidemic," *Journal of Applied Probability*, pp. 685–694, 1989.

[11] C. Nowzari, V. M. Preciado, and G. J. Pappas, "Analysis and control of epidemics: A survey of spreading processes on complex networks," *IEEE Control Systems*, vol. 36, no. 1, pp. 26–46, Feb 2016.

[12] H. Sayadi *et al.*, "Comprehensive assessment of run-time hardware-supported malware detection using general and ensemble learning," in *ACM Int. Conf. on Computing Frontiers*, 2018.

[13] H. Sayadi *et al.*, "Ensemble learning for hardware-based malware detection: A comprehensive analysis and classification," in *ACM/EDAA/IEEE Design Automation Conference*, 2018.

[14] H. Sayadi *et al.*, "Customized machine learning-based hardware-assisted malware detection in embedded devices," in *IEEE Int. Conf. On Trust, Security And Privacy In Computing And Communications*, 2018.

[15] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar 2014.

[16] P. H. J. Nardelli, P. Cardieri, and M. Latva-aho, "Efficiency of wireless networks under different hopping strategies," *IEEE Transactions on Wireless Communications*, vol. 11, no. 1, pp. 15–20, Jan 2012.

[17] C. Fleizach *et al.*, "Can you infect me now?: malware propagation in mobile phone networks," in *ACM W. on Recurring Malcode*, 2007.

[18] S. Hosseini, M. A. Azgomi, and A. T. Rahmani, "Malware propagation modeling considering software diversity and immunization," *Elsevier Journal of computational science*, vol. 13, pp. 49–67, Mar 2016.

[19] Intel, "Intel® 64 and ISA-32 architectures software developer's manual," 2016.

[20] M. R. Guthaus *et al.*, "MiBench: A free, commercially representative embedded benchmark suite," in *IEEE International Workshop on Workload Characterization*, 2001.

[21] J. L. Henning, "SPEC CPU2006 benchmark descriptions," *SIGARCH Comput. Archit. News*, vol. 34, no. 4, pp. 1–17, Sep 2006.

[22] B. Singh *et al.*, "On the detection of kernel-level rootkits using hardware performance counters," in *ACM on Asia Conference on Computer and Communications Security*, 2017.

[23] X. Wang *et al.*, "Hardware performance counter-based malware identification and detection with adaptive compressive sensing," *ACM Trans. Archit. Code Optim.*, vol. 13, no. 1, pp. 3:1–3:23, Mar 2016.