Network Traffic Analysis for Threats Detection in the Internet of Things

Mohammad Hammoudeh, *Senior Member, IEEE,* John Pimlott, Sana Belguith, *Member, IEEE* Gregory Epiphaniou, *Member, IEEE* Thar Baker *Member, IEEE* A.S.M. Kayes, *Member, IEEE* Bamidele Adebisi, *Senior Member, IEEE,* Ahcène Bounceur *Member, IEEE*

***Abstract*—As the prevalence of the Internet of Things (IoT) continues to increase, cyber criminals are quick to exploit the security gaps that many devices are inherently designed with. Whilst users can not be expected to tackle this threat alone, many current solutions available for network monitoring are simply not accessible or can be difficult to implement for the average user and is a gap that needs to be addressed. This paper presents an effective signature-based solution to monitor, analyse and detect potentially malicious traffic for IoT ecosystems in the typical home network environment by utilising passive network sniffing techniques and a cloud-application to monitor anomalous activity. The proposed solution focuses on two attack and propagation vectors leveraged by the infamous Mirai botnet, namely DNS and Telnet. Experimental evaluation demonstrates the proposed solution can detect 98.35% of malicious DNS traffic and 99.33% of Telnet traffic respectively; for an overall detection accuracy of 98.84%.**

***Index Terms*—Internet of Things, Network Traffic Analysis, Mirai Botnet, DNS Attacks, IoT Security.**

I. Introduction

RAPID developments in all areas encompassing the computing industry have been the dominant factor in phenomenal changes within the information systems landscape for home and enterprise environments alike. Within the last few decades, advancements in manufacturing, networking and distribution technologies have continuously challenged the position of what might be construed as the 'typical' computer. Devices are far more powerful and are often far more compact than their equivalents from even just a decade ago. With the clear benefits this combination of convenience and power can bring, small form factor systems have quickly become the preferred choice of system for many. The rise in use and development of compact interconnected devices has rapidly led way to the insurgence of the Internet of Things (IoT) [1].

The mass adoption of IoT technology has caught the attention of malicious entities hoping to benefit from exploiting a number of widespread security and access control issues inherent in many devices [2]. The nature of IoT devices results in maintaining a low level of user interaction and awareness in the state of the device itself. This, in conjunction with sparse updates from manufacturers results in prime targets for an attacker, commonly remaining unpatched and vulnerable for extended periods of time [3].

Often IoT focused malware exploit simple vulnerabilities that are rife within smart devices; a prime example being the Mirai botnet that surfaced in 2016. Mirai leveraged the Telnet protocol to infect and propagate by exploiting poor security controls in IoT devices that used the Busybox software suite. Once a device is infected, the botnet can then be controlled via a series of command and control servers (C2) that communicate through Domain Name System (DNS), a common method employed by many malware. However, botnets such as Mirai have built upon early iterations of similar malware like Bashlite to employ DNS [4] either as a direct attack vector to increase traffic, or to communicate with the owners' of the C2 with a directive to add 'threat mobility' effectively defeating many security monitoring systems that do not directly filter and analyse DNS queries.

Many Mirai derivatives such as Sora, Saikin and Akiru have surfaced since the original botnet began late in 2016; and many still employ the original attack, communication and propagation methods that made Mirai such a successful and devastating malware campaign. The OWASP IoT project maintain a list of prevalent security issues present in IoT devices[1]. Many of the main issues that allowed Mirai and other malwares to exploit IoT devices are covered by the latest iteration of the report, of which some of the top entries i.e. 'Weak, Guessable, or Hardcoded Passwords' and 'Insecure Network Services' would include services such as the Telnet protocol used by Mirai.

An effective DNS monitoring system in conjunction with or as part of an Intrusion Detection System (IDS) could assist in detecting malicious traffic to, or originating from the various domains associated with typical botnet C2. It is well understood that event logging and notification systems are an essential technical aspect within any modern effective information security management system [5]. IDS primarily focuses on analysis of generated logs created as a natural product of computer use, i.e., network logs generated through web requests and responses, and system logs generated by the operating system [6]. The challenge faced by classical

M. Hammoudeh J. Pimlott and B. Adebisi are with the Faculty of Science and Engineering, Manchester Metropolitan University, Manchester, United Kingdom, M1 5GD e-mail: m.hammoudeh@mmu.ac.uk).

S. Belguith is with the School of Science, Engineering & Environment, University of Salford, UK.

G. Epiphaniou is with the Warwick Manufacturing Group, University of Warwick, UK.

A. Kayes is with the Departement of Computer Science and Information technology, La Trobe University, Australia.

T. Baker is with the School of Computer Science, Liverpool John Moores University, UK.

A. Bounceur is with the Lab-STICC, University of Brest, France.

[1]https://www.owasp.org/index.php/OWASP_Internet_of_Things_Project

IDS systems within a smart environment is that a large majority of smart devices will not be able to effectively support host or client-side network-based IDS. The use of proprietary operating systems and specific embedded hardware architectures would potentially result in compatibility issues. In addition, the devices may be unable to provide the resources needed to effectively perform the monitoring functions [7]. Instead, the intensive processing and computing tasks of IDS systems should be off-loaded to other devices with appropriate resources, such as external servers, or gateway/proxy devices on the network. Traditional IDS systems are also not practical to setup by consumers who do not have advanced technical knowledge. Moreover, these IDS systems may be unable to detect IoT threats without significant customisation and fine-tuning of the underlying detection rules.

McDermott et al. [8] demonstrate that users are aware of the challenges related to the security and privacy of devices within their home network, but the majority simply lack the skills, technology and knowledge necessary to identify potential vulnerabilities. This gap between user awareness and technical ability is an issue that should be addressed to provide the average consumer with the means to accurately monitor their own devices for security issues.

This research presents an accessible cloud-based approach that can be used to monitor the security status of smart device networks in real-time, regardless of the user's technical knowledge and experience. The presented solution, called IoTMon, focuses on detecting attacks from the infamous Mirai botnet and its many derivatives which have been responsible for some of the largest Distributed Denial of Service (DDoS) attacks in recent history. The implementation of a network sniffing client to feed network traffic data to a cloud application is demonstrated and presented to provide a robust security monitoring solution.

The rest of the paper is organised as follow. Section II presents related work on threat detection approaches. The technical specifications of the proposed IoTMon platform are presented in Section III. Section IV describes IoTMon experimental evaluation setup and results. Section V concludes the paper and presents future work directions.

II. Related Work

There are a multitude of ways effective monitoring can be carried out using network traffic analysis. From elaborate honeypot systems to more traditional deep packet inspection and analysis on a web proxy or central node, a combination thereof could potentially result in a powerful, hybrid style IDS with potential to detect various sources of incoming and outbound malicious traffic. This section reviews some of common IDS systems and techniques.

Since the insurgence of IoT in recent years, various efforts have been made in developing bespoke network monitoring solutions aimed within the market, in addition to new research encompassing methods that can be employed to detect malicious behaviour by monitoring network traffic of IoT devices [9]. Multiple open-source tools, such as Snort, empower the user to install an effective IDS, however they often require significant technical expertise and knowledge to implement effectively and securely.

The majority of the open-source tools are packaged with a web interface or are enabled with remote administration capability upon direct install which requires further technical configuration. Most users would be hesitant to endeavour in attempting to configure or actualise their own detection system due to the absence of specialised skills or technical proficiency in implementing a traditional IDS solution. Even in commercial solutions such as BinaryEdge[2] and Shodan[3], there are also little to no user configurable options, and no ability to whitelist or blacklist specific network addresses. This makes the user solely reliant on the vendor to maintain and update their signature database. It is also plausible that an intermediary device controlling traffic allocation in place of a router could cause significant network configuration issues in some cases; as in the case of Fingbox[4] where DHCP allocation responsibilities are handed to the device. A critical concern is that routing all traffic through one intermediary device creates a single point of failure; if the monitoring device fails, likewise all devices connected through the node will fail.

Fingbox approaches IoT security monitoring with an alternative technique, rather than inspecting packets on the network for malicious traffic, this device monitors activity on the data-link layer for unrecognised devices attempting to connect to the network. Whilst this approach may be effective in detecting and preventing physical intrusion, the effectiveness of such a device may come into question if an already 'trusted' device is compromised.

Other commercial solutions similar to Fingbox are available such as Bitdefender's 'BOX'[5], which utilises Deep Packet Inspection (DPI) coupled with anomaly detection to identify threats. However, still most of these solutions require the user to route all traffic through the device, or the ones that do not, require the user to instead use their device in place of their own wireless access point, which has the potential to not perform as sufficiently. These platforms are only designed to function with the provided hardware devices, and most also do not allow the user to configure their own detection rules.

Pot2DPI [10] proposes to employ a honeypot solution in efforts to detect malicious traffic, by in a sense 'baiting' malicious traffic to a purposefully exposed endpoint. The honeypot approach is unique in a sense that the methodology can produce different results than traditional signature and anomaly-based detection by utilising poor security as an asset. Pot2DPI could potentially provide a framework for IDS whereby rules are created based on the traffic aimed at and originating from the honeypot, it can be assumed that traffic with unusual patterns taking place on the honeypot device is considered either malicious / nuisance traffic, and new signatures can be generated 'on the fly' in a reactive fashion. This solution could potentially provide a solid basis for a reactive Intrusion Prevention System (IPS); however, success of signature detection could be dependent on how the honeypot

[2]https://www.binaryedge.io
[3]https://www.shodan.io
[4]https://www.fing.com/products/fingbox
[5]https://www.bitdefender.co.uk/box/

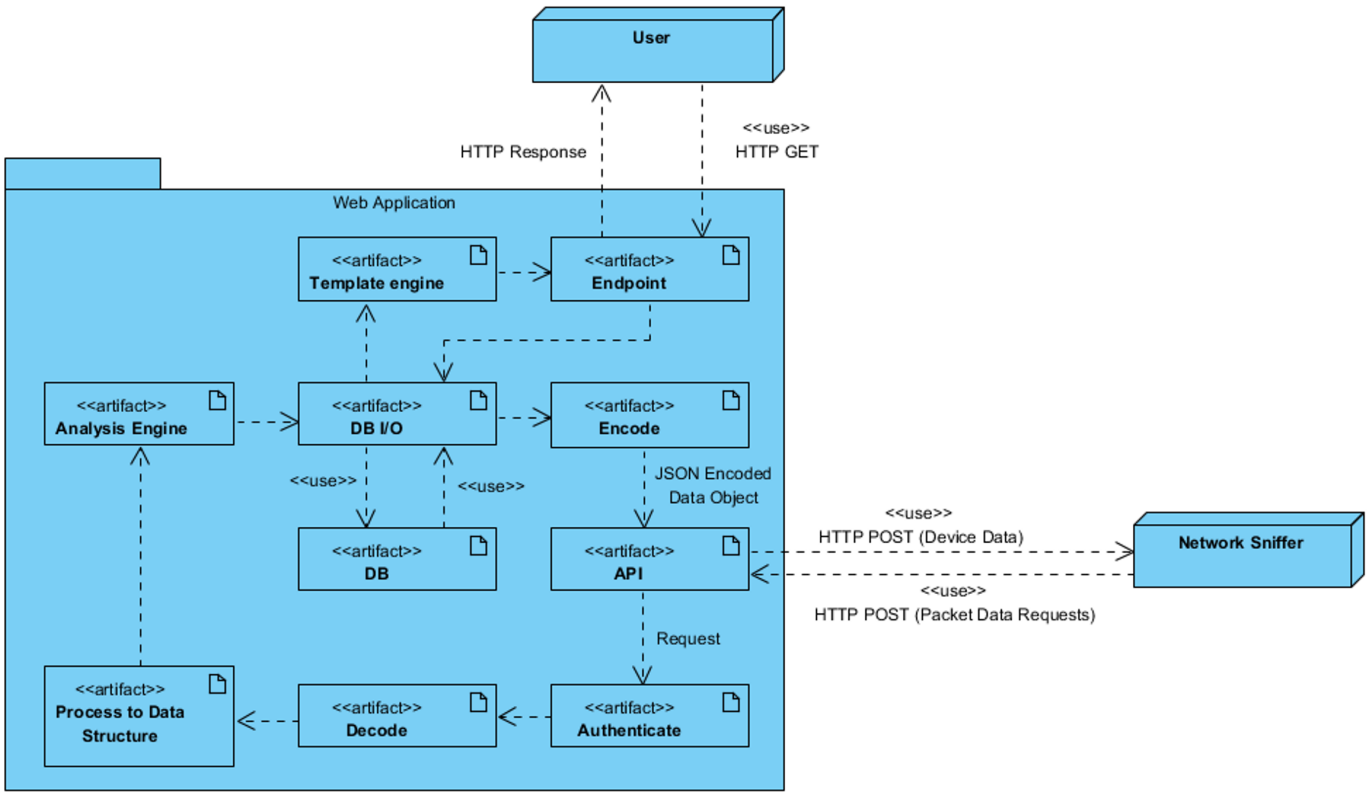

Fig. 1. An overview of IoTMon.

device is configured, and may also produce false positives without a fine tuned setup.

There are a multitude of ways effective monitoring can be carried out using network traffic analysis, from elaborate honeypot systems to more traditional deep packet inspection and analysis on a web proxy or central node, a combination thereof could potentially result in a powerful, hybrid style IDS with potential to detect various sources of incoming and outbound malicious traffic. However, signature-based detection is still a powerful tool, and could be more effective when such an emphasis is placed on user friendliness. With heuristic and anomaly-based detection the probability of identifying false positives increases. A high number of false positives could confuse the user, whereas a signature-based detection mechanism with a reliable signature dataset could produce more reliable results, and therefore the user can make a decision based on a higher degree of certainty.

Most of the established solutions do not cater specifically to an IoT network environment, and the recent products that do may still require the user to alter their network environment, potentially causing unexpected network issues. The commercial solutions available also do not provide a method of integrating other network traffic detection and packet inspection devices into the core analysis platform, and most do not allow the user to configure their own detection signatures and rules. The solution proposed in this paper aims to solve these challenges by creating a new cloud-based traffic monitoring platform, that abstracts the technical knowledge required to setup and maintain such a system away from the average user. A user friendly and effective traffic analysis platform could potentially assist to alleviate the growing risk in security within the IoT environment [11]. This platform will focus on detecting threats specific to a number of IoT devices, and implements an open API that will allow network level packet inspection devices to feed the service with data for analysis, regardless of manufacturer.

III. IoTMon Platform Details

In this section, the details of a new cloud-based *IoT* network traffic *M*onitoring, IoTMon for short, platform are presented. Figure 1 depicts IoTMon design, components, their interactions and communication protocols.

IoTMon abstracts the technical knowledge required to setup and maintain a network traffic analysis system away from the average user. It aims to provide a user-friendly platform to identify attacks on IoT devices that utilise the DNS and Telnet protocols, from initial infection and propagation using Telnet; to communication and obfuscation with botnet C2 servers using DNS. The solution consists of a cloud-based monitoring and analysis engine, in conjunction with a proof of concept network sniffer client to feed the cloud application with network data. The platform will focus on detecting threats specific to a number of IoT devices, and implements an open API that will allow network level packet inspection devices to feed the cloud-service with data for analysis, regardless of manufacturer. The following two subsections give the details of the cloud service, the network sniffer client and the designed open API.

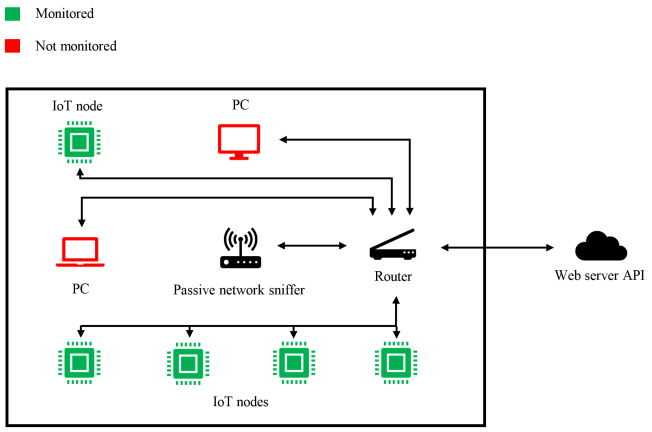

Fig. 2. AWPA-2 Protected WLAN and sniffer setup.

A. The Cloud-based Service Specifications

IoTMon packages all aspects of an analysis platform into a cloud-based service, which can be fed by any device via a special API. The developed service handles data analysis, processing and presentation stages entirely on the cloud. This includes the number of requests that have been sent to specific devices, and count summary of any malicious traffic that has been detected under on of the following categories 'Info', 'Warn' and 'Severe'. The three categories will correspond with incoming requests that are categorised into a 'risk group' via the back-end logic of the API when requests are sent to the service. The user is able to clearly identify any malicious traffic, along with key information that could assist in identifying malicious traffic and IoT device breaches. The protocol type for example, along with the payload, device and the timestamp at which the incident occurred.

The cloud service stores a database of signatures that can identify malicious network traffic; DNS records that point to known malicious domains for example, or Telnet passwords that are used to brute force access to accounts. The user is provided with a simple to use method of customising signature detection rule sets from within the web application. Alongside each signature will be an allocated 'risk rating' that designates which category the request would be placed into. Incoming requests will be analysed by the signature detection algorithm and if an incoming request matches with a signature, the database stores the request and updates the entry with its designated risk rating. The application will then pull these categorised requests, whichever is applicable, whenever the user accesses any of the category pages.

B. Passive Network Sniffing Client

A proof of concept network sniffing client to be paired with the web service is developed to demonstrate the viability of the IoTMon platform. The client allows the detection and data extraction from packets for the specific protocols focused on in this study, namely, DNS and Telnet. With this functionality the client will provide real-time network monitoring that works in collaboration with the web application by collecting network traffic from a IoT network, extracting all necessary information from the packets, and finally sending the data to the web service API for analysis.

Installing this sniffing client on each device to monitor the network traffic is feasible, however this would potentially require access to device firmware and configuration for each device being monitored. This approach would consume essential system resources and might also affect the originally intended operation of the device; this method would also fall outside most user's technical knowledge scope. However, an alternative approach could be to set up a passive wireless node that can capture all IoT device traffic. This approach differs from implementations discussed in Section II, as no traffic will actually be statically routed through the device. Instead, the sniffer can collect wireless traffic 'on the air' by utilising passive wireless packet sniffing techniques. Using a wireless network card in monitor mode, the device could be configured to perform the filtering, logging and collation of network data for analysis, passively collecting the required IoT wireless network traffic and sending this data to the web service for real-time analysis.

The client effectively runs a looping function that is triggered each time a packet on the network is detected, to reduce the number of packets being analysed and thus improving efficiency, a mac address filter has been implemented which is controlled via the cloud application. Each time the client starts up the client sends a request to the cloud application to pull a list of devices from the account. The client uses the MAC addresses to filter packets in the network; allowing an environment such as the one shown in Figure 2 whereby packets that are not addressed to or from defined devices will be ignored by the sniffer. This approach provides the user with the flexibility to dictate which devices to monitor on the network.

The processing algorithm is triggered by the incoming request API call and stores the result into the database, negating the need for multiple rounds of analysis. The initialisation routine is summarised in Algorithm 1.

Algorithm 1: Sniffer client initialisation sequence.

```
Begin:
    tet API key & network interface from arguments;
    retrieve user devices from web API;
    add devices to list;
if no devices in list then
    begin recording all traffic;
else
    apply traffic filter;
    begin recording filtered traffic;
end
```

The algorithm used by the client for the payload extraction is detailed in Algorithm 2. The code runs in a loop to build the sequence of characters that eventually create the password string.

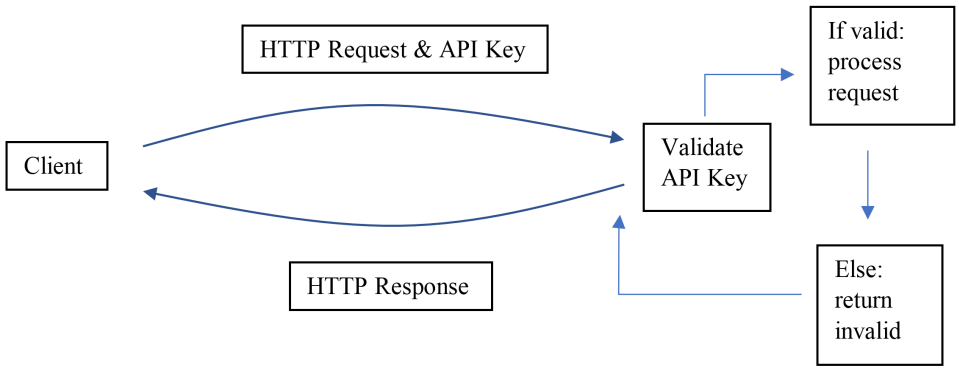

Fig. 3. An illustration of the API authentication cycle.

Algorithm 2: Sniffer client initialisation sequence.

```
Begin:
if packet is DNS: then
    extract query;
    extract request object & encode as JSON;
    send request containing query to API;
else
    if packet is TCP & source || destination port
     == 23: then
        extract payload;
        if payload is enter or submit;
            stop recording password;
        else if payload is login;
            stop recording password;
            join list of characters & send data to API;
        if recording password;
            if TCP sequence # ≠ previous packet;
                append payload character to list;
            else;
                ignore packet;
        if payload is the password prompt;
                begin recording payload to build
          character list
    else
        ignore packet
    end
end
```

C. Connectivity API

The connectivity API embedded into the web application covers two primary functions:

1) Allow sources to upload/feed data to the service to be analysed and subsequently presented to the user.
2) Allow sources to retrieve device data, i.e., device hardware MAC addresses and names, for filtering network traffic capture.

Both API endpoints are protected via a randomly generated API key that is created on account registration. The user can access the API key via a cloud dashboard and uses this to connect external devices/services to the application; each API key is unique to a single user account, allowing the incoming and outgoing data to be associated to a specific user.

API requests are validated by checking the provided API key in each POST request against an existing user account. All API routes expect a key to be provided as part of the HTTP request header. An invalid or null key in the header will result in an invalid response from the web server and that request will not be processed by the application. API requests are validated by checking the provided API key in each POST request against an existing user account. All API routes expect a key to be provided as part of the HTTP request header. An invalid or null key in the header will result in an invalid response from the web server and that request will not be processed by the application. The API authentication cycle is depicted in Figure 3

Utilising the standard JSON data format, the API can be accessed and interacted with by any compatible devices contingent that both the correct format, and valid API keys are provided. The API uses the parameters in Table I as arguments to pass and exchange the incoming data to be analysed by the processing routines.

IV. Evaluation

This sections aims to assess the real-world effectiveness of IoTMon's detection mechanisms by utilising the network-sniffing client as a testing device to monitor and inspect network traffic, sending packets to the cloud API to be analysed in real-time. A basic attack scenario is emulated, followed by a thorough round of testing for both the Telnet and DNS protocols separately.

Inspiration from the Model, View, Controller (MVC) design pattern is utilised to ensure the application itself and the development process will remain modular. The Jinja2 template framework primarily controls the 'View' portion of the application. The template language is a powerful tool that converges the presentation layer with inherited python code, however efforts are made to keep any large data processing within the main python codebase, rather than embedded in the HTML files to separate the logic of the backend and frontend code bases. The 'Model' portion is primarily controlled via a series of dedicated SQL queries and calls in the application; similarly, the 'Controller' section will consist of back-end server code and algorithms to process the data returned by

TABLE I
JSON PARAMETERS OF REQUEST OBJECTS.

Parameter	Description	Example
protocol	Request protocol type	protocol:Telnet
alertType	Query or action type	alertType: TelnetLogin-Success
timeOfRequest	Request time	timeOfRequest: 2019-01-25 17:08:35
payload	Request data payload	payload: administrator
deviceAddress	Device MAC address	deviceAddress: 5c:96:8f:90:2c:fd

the 'Model' code. Using the MVC pattern creates an abstract layer between functions in the application that improves extensibility of the program which will aid the development process when the codebase becomes sizeable.

The first test simply comprises of a login to the telnet service on a host machine, followed by sending a small number of DNS requests to known malicious domains from the compromised host. The test aims to evaluate whether both the sniffer and web application can operate in tandem to effectively detect the compromised telnet login and malicious DNS requests. After this has been validated, to further test the robustness of the platform, both Telnet and DNS will be rigorously assessed separately. DNS detection rates will be tested by generating a large batch of DNS requests to known malicious domains that have been stored in the cloud application signature database. The malicious domains have been sourced from the AlienVault open threat exchange, which have been identified as Indicators of Compromise (IoC) of Mirai and derivative malwares [12]. The telnet login detection was also be rigorously tested, by using the well-established offensive security tool, 'Ncrack' in an attempt to brute-force the telnet account password with a small password list multiple times in succession. All passwords being used will be credentials sourced from the studies of Elzen and Heugten [13] whom obtained the list of passwords used in the Mirai attacks by reverse engineering the Mirai source code. The sniffing node is setup in monitor mode where no special traffic conditions are imposed. Finally, the sniffer node is placed nearby to the wireless access point.

All of the above testing was carried out using a live hosted version of the cloud application on AWS infrastructure, thus providing a realistic insight into the performance metrics of the application. Ultimately, the testing has been specifically designed to allow realistic evaluation of the product in a typical home network environment.

The obtained results demonstrate that the system can detect malicious DNS traffic and Telnet login attempts with success, with a combined detection accuracy of 98.84%. The solution successfully captured the initial test attack scenario, showing one successful telnet login, accompanied by the small batch of DNS requests sent from the compromised host.

Testing the DNS and Telnet detection capabilities in depth revealed promising results. First, a batch of 790 DNS requests in total were executed on the 'compromised' system, at the end of the session the results within the web application showed a total of 803 detected malicious requests, showing a small discrepancy between the expected number, i.e., ground truth, and the actual result. A total of 13 false positive results were detected, resulting in an error percentage of 1.65%, or alternatively a measurement of 98.35% detection accuracy.

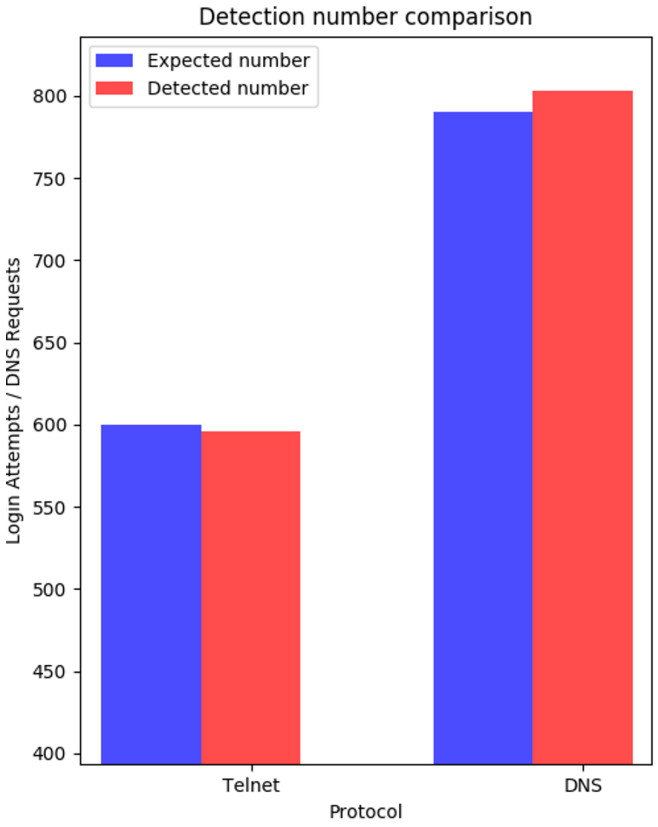

Fig. 4. Comparison of detected requests vs expected number per protocol.

Testing of the Telnet detection functionality was carried out by using the 'Ncrack' login attack tool in single thread mode. Ncrack was launched against the local telnet server using a 10-entry password list of which one entry was the correct password; a similar method as utilised in the Mirai malware [14], however the same account and thus password was used in this scenario. In this session a total of 400 login attempts were made by Ncrack; with 40 successful logins. The obtained results showed a total of 39 detected successful logins out of 40; however, the precision of the incorrect login detection rate could not be accurately measured; due to intermittently sending duplicate requests to the cloud API when an incorrect login was detected; after fixing this issue, the Telnet detection was tested once again demonstrating better results.

In the second round of testing, a total of 600 login attempts were made by Ncrack, making 60 successful logins; and 540 unsuccessful login attempts. The statistics demonstrate promising results, presenting a total of 60 successful login attempts

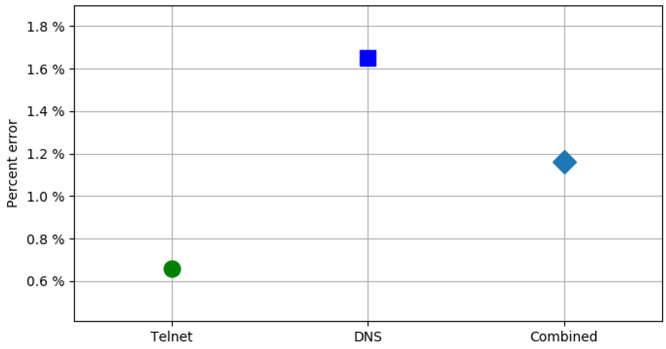

Fig. 5. Comparison of percent error rates.

TABLE II
THE PERFORMANCE OF IOTMON AGAINST VARIOUS MACHINE LEARNING TECHNIQUES.

Technique	**Accuracy**
CART	80.3%
Multi-layer perception	87.41%
Naïve Bayes	87.56%
KNN	88.4%
SVM	89.52%
C4.5	92%
Random Forest	92.67%
IoTMon	98.84%
HIDS	99.97%

identified, with 536 identified unsuccessful login attempts. Culminating in a total true positive count of 596 and false negative count of 4; resulting in a 0.66% error percentage rate, or alternatively to be measured as 99.33% detection accuracy, see Figure 4.

Combining the percent error rates of the telnet and DNS testing results in a combined error rate of 1.16% or detection accuracy of 98.84% (see Figure 5; demonstrating that the platform can reliably detect malicious traffic from both protocols with a high degree of precision.

We compare IoTMon's performance against seven complementary baseline methods. Namely, we use the publicly available evaluation results in [15] for Classification And Regression Tree (CART), multi-layer perception, Naïve Bayes, K-Nearest Neighbour (KNN), Support Vector Machine (SVM), C4.5 and Random Forest on the Bot-IoT dataset. We also compare against HIDS which combines the advantages of signature- and anomaly-based intrusion detection systems [15]. Table IV summarises the performance evaluation results in terms of attack detection accuracy of the above listed machine learning techniques, HIDS and IoTMon. The results show that IoTMon provides comparable results to HIDS without the complexity and overhead associated with machine learning.

V. CONCLUSION AND FUTURE WORK

IoTMon provides a flexible, open platform alternative for IoT device security monitoring. Technical proficiency is not a strict limiting factor for the consumer and could lead way to a safer environment empowering the average user with the means to take appropriate action in protecting their network. Experimental evaluation results demonstrate that IoTMon could be a viable solution for effective intrusion detection for IoT devices with reliable detection accuracy.

Whilst the evaluation has provided great insight into areas where IoTMon has succeeded; there is potential room for improvement in the future. Creating a fully mobile responsive dashboard would be a valuable addition to the works conducted thus far, as it would provision users with the ability to monitor their networks remotely, away from a traditional computer system. The use in mobile devices continues to rise, as such supporting this platform would be a valuable and justified addition to the product, whilst remaining in touch within the context of IoT. additionally, incorporating alternative detection methodologies such as heuristic, anomaly and honeypot-based detection could also be explored to work in conjunction with the signature-based technology utilised within this work; with the potential to further enhance the attack detection capabilities.

REFERENCES

[1] R. Ande, B. Adebisi, M. Hammoudeh, and J. Saleem, "Internet of things: Evolution and technologies from a security perspective," *Sustainable Cities and Society*, p. 101728, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210670719303725

[2] S. Belguith, N. Kaaniche, M. Hammoudeh, and T. Dargahi, "Proud: Verifiable privacy-preserving outsourced attribute based signcryption supporting access policy update for cloud assisted iot applications," *Future Generation Computer Systems*, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X1930130X

[3] I. Yaqoob, E. Ahmed, M. H. ur Rehman, A. I. A. Ahmed, M. A. Al-garadi, M. Imran, and M. Guizani, "The rise of ransomware and emerging security challenges in the internet of things," *Computer Networks*, vol. 129, pp. 444 – 458, 2017, special Issue on 5G Wireless Networks for IoT and Body Sensors. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128617303468

[4] A. Marzano, D. Alexander, O. Fonseca, E. Fazzion, C. Hoepers, K. Steding-Jessen, M. H. P. C. Chaves, I. Cunha, D. Guedes, and W. Meira, "The evolution of bashlite and mirai iot botnets," in *2018 IEEE Symposium on Computers and Communications (ISCC)*, June 2018, pp. 00 813–00 818.

[5] J. Chaudhry, A.-S. K. Pathan, M. H. Rehmani, and A. K. Bashir, "Threats to critical infrastructure from ai and human intelligence," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 4865–4866, 2018.

[6] M. Shakil, A. Fuad Yousif Mohammed, R. Arul, A. K. Bashir, and J. K. Choi, "A novel dynamic framework to detect ddos in sdn using metaheuristic clustering," *Transactions on Emerging Telecommunications Technologies*, 2019.

[7] C. Iwendi, P. K. R. Maddikunta, T. R. Gadekallu, K. Lakshmanna, A. K. Bashir, and M. J. Piran, "A metaheuristic optimization approach for energy efficiency in the iot networks," *Software: Practice and Experience*, vol. n/a, no. n/a, 2020.

[8] C. McDermott, J. Isaacs, and A. Petrovski, "Evaluating awareness and perception of botnet activity within consumer internet-of-things (iot) networks," *Informatics*, vol. 6, no. 1, p. 8, Feb 2019. [Online]. Available: http://dx.doi.org/10.3390/informatics6010008

[9] I. Ghafir, V. Prenosil, J. Svoboda, and M. Hammoudeh, "A survey on network security monitoring systems," in *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, Aug 2016, pp. 77–82.

[10] V. Martin, Q. Cao, and T. Benson, "Fending off iot-hunting attacks at home networks," in *Proceedings of the 2nd Workshop on Cloud-Assisted Networking*, ser. CAN '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 67–72. [Online]. Available: https://doi.org/10.1145/3155921.3160640

[11] S. Walker-Roberts, M. Hammoudeh, O. Aldabbas, M. Aydin, and A. Dehghantanha, "Threats on the horizon: understanding security threats in the era of cyber-physical systems," *The Journal of Supercomputing*, Oct 2019. [Online]. Available: https://doi.org/10.1007/s11227-019-03028-9

[12] AlienVault, *Mirai is Attacking Again*, 2020 (accessed January 26, 2020). [Online]. Available: https://otx.alienvault.com/pulse/5a787f34a0cfdc7fb5945d8b

[13] I. van der Elzen and J. van Heugten, "Techniques for detecting compromised IoT devices," Master's thesis, University of Amsterdam, the Netherlands, 2017.

[14] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, and et al., "Understanding the mirai botnet," in *Proceedings of the 26th USENIX Conference on Security Symposium*, ser. SEC'17. USA: USENIX Association, 2017, p. 1093–1110.

[15] A. Khraisat, I. Gondal, P. Vamplew, J. Kamruzzaman, and A. Alazab, "A novel ensemble of hybrid intrusion detection system for detecting internet of things attacks," *Electronics*, vol. 8, no. 11, 2019.