IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 25, 2022, accepted May 9, 2022, date of publication May 17, 2022, date of current version May 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3175886

An AI-Powered Network Threat Detection System

BO-XIANG WANG, JIANN-LIANG CHEN, (Senior Member, IEEE), AND CHIAO-LIN YU
Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei 106, Taiwan

Corresponding author: Jiann-Liang Chen (lchen@mail.ntust.edu.tw)

ABSTRACT The work develops a network threat detection system, AI@NTDS, that uses the behavioral features of attackers and intelligent techniques. The proposed AI@NTDS system combines data analysis, feature extraction, and feature evaluation to construct a detection model, which supports a more straightforward strategy by which the operating system or its operators can defend against network attacks. The Linux system interaction information of SSH (Secure Shell) and Telnet are obtained from the Cowrie Honeypot and labeled according to Enterprise Tactics of MITRE ATT&CK to ensure dataset credibility. The proposed AI@NTDS system has three levels, depending on the attacker's attacks and the user's risk of damage. Fifty-two features are used to detect the network threat level. The features contain message-based features for all kinds of Linux operating instructions, host-based features for all types of information in the network connection process, and geography-based features are related to the attacker's location. AI-based algorithms LightGBM, Random Forest and the K-NN algorithm are used to verify the identification of the custom features. Finally, the detection model that is trained using the best combination of features is used to predict the test dataset. The accuracy of the proposed AI@NTDS system reaches 99%, 95.66%, and 94.08% with the LightGBM, Random Forest, and K-NN algorithms, respectively. The mutual dependencies of features and network threats are evaluated. Results of a performance analysis reveal that the proposed AI@NTDS system has an accuracy of 99.20% and an F1-score of 99.80%. It is superior to existing detection mechanisms, which it outperforms by 4% and 1% in accuracy and F1-score, respectively.

INDEX TERMS Honeypot, intent analysis, machine learning, remote shell access, threat detection.

I. INTRODUCTION

The Internet of Things (IoT) is utilized in various industries, and more IoT devices are being connected to the Internet every day. By 2021, 35 billion IoT devices had been installed worldwide [1]. The global volume of data is increasing exponentially as the IoT grows. Remote controls of the devices are frequently used as they are convenient and support resource sharing in the IoT environment. Most IoT devices are based on Linux, and they are remotely controlled using Telnet or SSH. The password authentication strategy is used to protect these remote devices. However, hackers can use brute force to search for passwords in insecure situations. Hackers can break into a system through remote connections such as Secure Shell (SSH) or Telnet. A hacker who enters a control system will perform reconnaissance or download and execute malicious files to obtain system permissions and, ultimately, to steal sensitive information from an enterprise or organization. Most devices communicate using SSH protocol remote access services. Since this protocol provides encrypted communication between the SSH client and the server, an attacker that is connected to the server can execute various malicious services.

Venafi, Inc. collects real-world examples of SSH threats [2]. For example, Sony Pictures was hacked in 2014 and SSH keys were stolen, leading to leaks of executive salaries and copies of unreleased Sony movies. The 2019 Kinsing Malware included several shell scripts that download and install, remove, or reinstall various services and programs. The 2020 Kaiji malware detected poorly configured SSH services and performed a brute force attack [3]. The above examples show that SSH attacks involve various behaviors. Therefore, SSH security is critical and user access to remote systems must be carefully monitored. Commands that are executed by a remote connection must be analyzed.

In this study, AI-powered techniques are used to solve the command-based content problem and design a network threat detection system, AI@NTDS. Since an enormous amount of information is collected daily, the manual defense of the remote connection threats may cause an irreversible situation. The malicious command dataset for AI Model training is

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

collected and organized by the Honeypot. Most importantly, the problem of detecting malicious commands is solved herein.

A. PROBLEM STATEMENT

Many researchers have presented solutions to protect users against the command-line-based threat. The main task that will be addressed in this work is the detection of the hacker's malicious intent; 52 features will be provided for the analysis of the AI model. These include message-based, host-based, and geography-based features.

B. CONTRIBUTIONS

This work contributes to the field by developing an AI-powered network threat detection system, AI@NTDS, which has three levels. The system provided 52 features for the AI-based threat detection datasets. The three main types of features are message-based, host-based, and geography-based. A feature importance analysis demonstrates that Message_Length, Execution_File, and Received_Size features for the malicious behavior are more critical than other features. The features proposed herein are effective in detecting remote network connection threats. The performance analysis results herein were much better than those in other studies, revealing that the model in this study had an accuracy of 99.2% and a performance of F1-score of 99.8%.

II. RELATED WORK

The section will review the latest SSH-based intrusion systems, techniques, and experiments. Descriptions of the experiments have been published in different scientific articles, and various threats have been detected.

A. THREAT INDICATOR WITH THE HONEYPOT

Fraunholz *et al.* [4] found that criminal activity on the Internet is becoming more sophisticated. Traditional information security technologies can barely cope with recent trends in such activity. In this investigation, several Honeypots are combined to form a honeynet. The Honeynet ran for 222 days, and 12 million attack attempts were captured. The captured data are examined and evaluated herein. The experimental results can identify and quantify the dependences and distributions of the data. New threats are constantly emerging, so capturing the features of attacks and analyzing them effectively is essential [5]. Several Honeypot sensors were deployed to monitor and study (the attackers' behavior. Honeypots type are in Cowrie, Dionaea, and Glastopf, in Linux hosts, Windows host, and web application environments. The above Honeypots attract various attacks from different environments.

Kumar *et al.* [6] improved the deployment and maintenance of tight tanks for various IT systems and intensive resource requirements. Security researchers and security companies extensively use Honeypot because it traps and understands attackers' tools and strategies. The deep learning-based analysis that is inspired by neural networks is integrated into classifying threat events. Jason *et al.* introduced a tool for evaluating Honeypot [7]. Honeypots are used to capture traces of malicious activity. They can be used to study an attacker's behavior, but they can be challenging to implement and maintain. This study outlines a complete Honeypot design, conducts experiments in data, and presents results thus obtained. The evaluating tool's design is outlined, and the results are provided as quantitative calibration data.

B. ANALYSIS OF ATTACKERS' BEHAVIORS BASED ON SSH SESSIONS

Following the above definition of Honeypot, this subsection will discuss the use of the information collected for analysis of the collected information. The definition and analysis of the behavior in Honeypot using previously developed research methods are described.

Esmaeil *et.al.* proposed a Honeypot technique to investigate violent SSH attacks on academic networks [8]. The most common attack is the strong guess-password attack that targets SSH, FTP, and Telnet servers. Experimental results demonstrate that preset lists of user names and passwords are widely shared and form the basis of violent attacks. Valli *et al.* [9] used the Kippo SSH Honeypot system to identify the activity in the Honeypot. The system runs on the same hardware and software configuration as above. Data over 75 days were collected as experimental data. An analysis yields the attackers' behaviors and patterns. The experimental results show that the number and range of attacks are different so that the content can be further discussed.

Kambourakis *et al.* [10] discusses the current state of botnets affecting the Internet of Things and the reasons for causes of the success of attacks. They provided detailed information on the operating principles of malware in the Internet of Things, examined their interrelationships, and proposed preventative strategies against malware. Critical steps concerning the operation and communication of botnets have been proposed and six sets of features of Mirai botnets have been identified [11]. That study used the above features to secure IoT devices and protect Internet infrastructure from destructive distributed denial-of-service attacks.

Bajtos *et al.* [12] observed botnets and described the behavior of the first two stages of their life cycle, which are initial infection and secondary infection. They resolved identified the behavioral attributes in each stage and designed a model to determine whether a threat is a botnet. They found that some network sessions and credential guesses are easily collected and usable attributes of the features in profiling threat agents.

C. DETECTION OF ATTACKS USING AI TECHNIQUES

Laurens *et al.* presented a stream-based SSH attack detection system, SSH Cure [13]. The system uses a machine-learning algorithm and observes network traffic to detect attacks and find their targets in real time. A prototype, including a graphical user interface, was implemented as a plugin for the popular NfSen monitoring tool. Sadasivam *et al.* [14] grouped SSH attacks into two types - severe and non-severe. A severe

attack is any attack that follows the successful corruption of an SSH server. A non-severe attack is any attack that fails. This study presents 14 features that are used in the real-time classification of attacks using machine learning algorithms. Bajtoš *et al.* [15] focused on the infection by botnets that are grouped into nine series by the features of the collected samples. They experimentally identified dependencies between commands and directories. Arifianto *et al.* [16] proposed the SSH Honeypot architecture that uses an Intrusion Detection System. Their work addressed SSH service attacks by observing the number of login attempts between two Honeypots, and the attack risk was determined using category weights and port scanner detection results.

Shrivastava *et al.* [17] used virustotal.com and the relevant literature to categorize attacks into four classes - malicious, SSH, XOR DDoS, and spying. Dumont *et al.* [18] proposed a fully implemented binary classifier that used machine learning algorithms to differentiate between malicious and benign shell commands. The classification results thus obtained were combined with the results obtained using the 1-Command and n-Command classifiers. Udhani *et al.* [19] analyzed an SSH-based Honeypot to identify automated and human attackers. The method used the number of requests, the target of the attack, the frequency of requests, and passwords.

Lee *et al.* [20] used the packet length in an SDN switch in deep learning models to identify anomalous and malicious packets. J. M. Jorquera *et al.* proposed a method for classifying threats that was based on Linux command's property [21]. They used machine learning algorithms to identify and classify an attacker's malicious intentions in executing a cyber threat based on the severity of the command. Lingenfelter *et al.* [22] showed that the Cowrie Honeypot is an effective system for collecting samples of malicious sessions. The same loader session can be found by finding the Edit Distance between command sequences Garre *et al.* [23] designed an approach to detecting botnets that is based on an SSH-based Honeypot. Their dataset contained 93 functions, including commands, session status, and network statistics. They used the random forest algorithm in experiments.

Wang *et al.* [24] used their previously obtained research results to evaluate the threats and applied AI technique to the Zenodo CyberLab Honeypot dataset and compared the LightGBM algorithm with random forest and K-NN algorithms. With different feature sets in the SSH-based Honeypot for the cyber-threat intelligence system. The system was found to be superior to existing detection methods.

D. NETWORK INTRUSION DETECTION SYSTEM DESIGN WITH AI

Alzahrani, A.O. *et al.* designed a machine learning-based intrusion detection system for software-defined networking (SDN). They used 41 features from NSL-KDD datasets for multi-class classification, and detected four kinds of attack DDoS, PROBE, R2L, and U2R [25]. S. Iranmanesh *et al.* proposed a heuristic distributed scheme (HIDE) to validate the falsification of traffic data. Their calculations were based on a homogeneous semi-Markov process that predicted the accuracy of mobility patterns. They used a cloudlet with a weight factor to determine whether a vehicle is malicious [26].

M. Sewak *et al.* sought to fill the gap between AI-based accomplishments and a comprehensive review of the cyber security threat landscape. They proposed and reviewed a machine-learning solution for threat detection and endpoint protection using deep reinforcement learning [27].

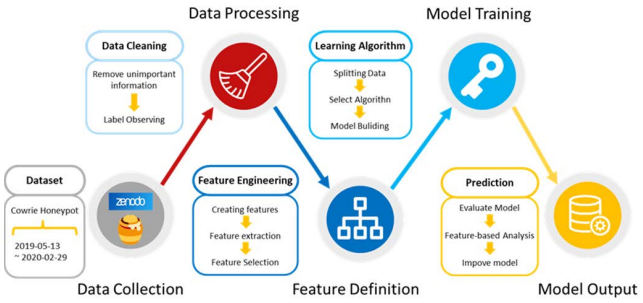


FIGURE 1. Proposed AI@NTDS system architecture.

III. PROPOSED NETWORK THREATS DETECTION SYSTEM- AI@NTDS

An intelligent threats detection system, called the AI@NTDS system, is designed to investigate network threats and analyze them using specific features and algorithms, primarily for SSH sessions. This section describes the automatic data acquisition process and defines the features. A multilabel classification model will also be introduced.

A. SYSTEM ARCHITECTURE

The proposed AI@NTDS system architecture that is shown in Figure 1 has five parts, which perform data collection, data preprocessing, feature-based analysis, model training, and model output.

The dataset of Cowrie Honeypots was obtained from the CyberLab Honeypot-Zenodo [28]. The various attack features are identified from variations among attacks. Fifty-two features were extracted from the Cowrie Honeypot dataset and then grouped into message-based, host-based, and geographic-based features. The data preprocessing part ensures that the labels and contents in the samples are correct. The algorithm is used to evaluate the importance of various feature combinations. To ensure the stability of an AI model and prevent overfitting, the model training process should not learn too closely with the result of the training dataset. The model is validated during the training process. The performance of the presented model is determined at various times, and the results thus obtained are presented in the following section.

B. DATA COLLECTION

The Cyberlab Honeypot collected attackers' data from June 2019 to February 2020 for use in this study. Cowrie Honeypots, with approximately 50 nodes mostly at univer-

FIGURE 2. Flow chart of data collection.

TABLE 1. Enterprise tactics in MITRE ATT&CK.

ID	Name	Description
TA0043	Reconnaissance	The adversary is trying to gather information they can use to plan future operations.
TA0042	Resource Development	The adversary is trying to establish resources they can use to support operations.
TA0001	Initial Access	The adversary is trying to get into your network.
TA0002	Execution	The adversary is trying to run malicious code.
TA0003	Persistence	The adversary is trying to maintain their foothold.
TA0004	Privilege Escalation	The adversary is trying to gain higher-level permissions.
TA0005	Defense Evasion	The adversary is trying to avoid being detected.
TA0006	Credential Access	The adversary is trying to steal account names and passwords.
TA0007	Discovery	The adversary is trying to figure out your environment.
TA0008	Lateral Movement	The adversary is trying to move through your environment.
TA0009	Collection	The adversary is trying to gather data of interest to their goal.
TA0011	Command and Control	The adversary is trying to communicate with compromised systems to control them.
TA0010	Exfiltration	The adversary is trying to steal data.
TA0040	Impact	The adversary is trying to manipulate, interrupt, or destroy your systems and data.

sities and companies in the European Union and the United States were used. Each file in the dataset is based on reports of daily intrusions. Sessions are grouped according to the attacker invades and leaves. Each group of sessions contains various events and explicit intentions. This goal of this work is to reduce the complexity and automatically to collect results daily. This system automates the process by applying the concept of a crawler. The data collection program automatically decompresses and converts the extracted JSON file into a CSV file. Figure 2 presents the flow chart.

C. DATA PROCESSING

Data processing firstly removes irrelevant information from the dataset to ensure data quality. The first step in this process is the removal of data associated with failed intrusions. Empty fields are deleted to save storage space and increase computing efficiency. Then, the cleaned data are labeled in a manner consistent with Enterprise Tactics in MITRE ATT&CK Enterprise Tactics comprise 14 groups of Tactics, of which those used herein are indicated below.

Table 1 presents the results obtained using the indicated Tactics. Nine tactics in the dataset are labeled with ''no intention''. They include No Action, Execution, Persistence, Privilege Escalation, Defense, Credential Access, Discovery, Command and Control, Impact. These tactics are associated with three malicious levels based on severity. Level 1 refers to actions that may damage the system, such as the execution of malicious files that stop the system. It is the most dangerous and malicious intention for a system, such as when a hacker inputs the command ''kill'', or ''rm'', or executes some unknown executable binaries. Level 2 refers to setting file permissions for personal accounts. For example, a hacker may input the ''chmod'' command or ''chattr'' command to change the file permission. Level 3 refers to the absence of

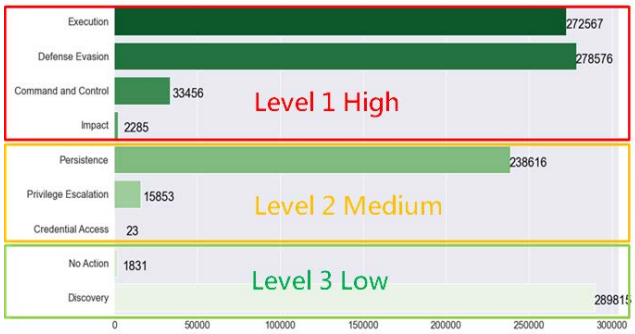


FIGURE 3. Data distribution of aggressive behavior.

TABLE 2. Training, validation and test datasets.

Dataset Purpose	Quantity	Collection Time	Proportion
Training	184463 samples	2019-06-04 ~ 2019-12-31	66%
Validation	46116 samples		11%
Test	68088 samples	2020-01-01~ 2020-02-29	23%

action or scouting actions. For example, if a hacker inputs a command like ''cat /etc/passwd'' and ''lscpu'' to obtain system information, the command will be assigned to level 3.

Figure 3 presents the tactics distribution of labeled data. Defense Evasion is the most common tactics at Level 1. The attacker's purpose is not to leave records of the removal of downloaded programs, to destroy files, and to obfuscate the system. Malicious programs are commonly used to run scripts to set permissions and perform other actions that are typical of attackers in Honeypots. The most common tactic in Level 2 is Persistence. The attacker's purpose is to escalate privilege to an account. When a connection break occurs, an attacker maintains access to the system to support the malicious operations, or change the system configuration. The figure below shows that the most common attack following login at Level 3 is Discovery because when an attacker accesses the system, the first task is always to perform reconnaissance. The distribution of the tactics of an attacker when he enters a Honeypot can be identified from statistical data.

The data in our work were collected from the 4th of June, 2019, to the 29th of February, 2020. The data were automatically collected using a web crawler, which grabbed 153,665,690 samples. After the invalid samples were removed, 298,667 valid sample entries remained to undergo the following process. The data from 4th June 2019 to 31th December 2019 formed the training dataset, while those from the 1st January 2020 to the 28th February 2020 formed the test dataset. Of the training data, 15% were allocated to the validation dataset that was used to evaluate the model. Table 2 presents the applications of the split dataset.

D. FEATURE DEFINITION

This subsection introduces and describes in detail 52 groups of features. These groups were divided into message-based, host-based, and geography-based types groups, as shown in Table 3. The algorithms that are used for machine learning

TABLE 3. Fifty-two features.

Features Set : 52 Features					
F1	*Keyword_bash*	F18	*Keyword_cat /etc (Get_sys_info)*	F35	***Count_Hex***
F2	*Keyword_shell*	F19	*Keyword_uname (Get_sys_info)*	F36	***Count_url***
F3	*Keyword_exit*	F20	*Keyword_wc (Get_sys_info)*	F37	***Message_length***
				F38	***Messages / sec***
F4	*Keyword_help*	F21	*Keyword_crontab (Get_sys_info)*	F39	*Protocol*
F5	*Keyword_passwd [user] (Set_account)*	F22	*Keyword_w (Get_sys_info)*	F40	*Src_Port*
F6	*Keyword_chpasswd (Set_account)*	F23	*Keyword_ps (Get_sys_info)*	F41	*SSH_Client_Version*
F7	*Keyword_useradd -[suffix] (Set_account)*	F24	*Keyword_free (Get_sys_info)*	F42	*Username*
F8	*Keyword_. [file] (Execution_file)*	F25	*Keyword_lscpu (Get_sys_info)*	F43	*Password*
F9	*Keyword_sh [file] (Execution_file)*	F26	*Keyword_nproc (Get_sys_info)*	F44	*Duration*
F10	*Keyword_./ [file] (Execution_file)*	F27	*Keyword_uptime (Get_sys_info)*	F45	***Received_Size (AVG.)***
F11	***Keyword_perl [file] (Execution_file)***	F28	*Keyword_wget (Network_connect)*	F46	*File*
F12	***Keyword_python [file] (Execution_file)***	F29	*Keyword_tftp (Network_connect)*	F47	*Continent_Code*
F13	***Keyword_/bin/ [file] (Execution_file)***	F30	*Keyword_scp (Network_connect)*	F48	*Country_Name*
F14	*Keyword_chmod .[file] (Set_permissions)*	F31	*Keyword_ping (Network_connect)*	F49	*Region_Name*
F15	*Keyword_sudo su (Set _permissions)*	F32	*Keyword_kill (Shutdown_action)*	F50	*City_Name*
F16	*Keyword_rm [file] (Delect_record)*	F33	*Keyword_reboot (Shutdown_action)*	F51	*Longitude*
F17	*Keyword_history -[suffix] (Delect_record)*	F34	***Count_base64***	F52	*Latitude*

focuses on the weights of feature data, so feature extraction is crucial. In addition, the features proposed by the authors are the use of red font marks.

1) MESSAGE-BASED FEATURES

Features of F1 to F4 are observed an invalid instruction is generated. Features F5 to F7 are account-related instructions. Features F8 to F13 are instructions concerning executing files. Features F14 and F15 are directives concerning to promote permissions. Features F16 and F17 are instructions to delete the attack history. Features F18 to F27 are information about the attacker observing the system. Features F28 to F31 are the network-related instructions to download or transmit files. Features F32 and F33 are instructions that affect the state of the system. Features F34 and F35 are used to satisfy the requirements of an attack by obfuscation. Features F36 and F37 calculate the number of URLs in messages and the length of each messages, respectively. F38 is the number of characters entered per second. This study propose eight features. Features F11 to F13 are used to calculate the number of keywords *perl [file]*, *python [file]*, and */bin/ [file]*, respectively. The functions of Features F35 to F36 were given above. Features 34, 37 and 38 are described below.

F34 Count_base64: An attacker will always use base64 encoding to obfuscate malicious behavior. One of the most common features is that attack scripts are encoded and decoded at execution time. Therefore, this feature is used to determine whether the command Base64 is present in messages. Figure 4 indicates the results of comparison between Base64 encoding and decoding.

F37 (Message_length) and F38 (Messages/sec): These two features are used to calculate the total length of a message. A message without any intention is shorter in total than one with a particular intention. The essential purpose of these kinds of commands is to evade detection or to achieve multiple goals. The number of characters entered per second is used to determine whether the attacker is a robot or script execution.

Table 4 presents the details of the message-based features.

FIGURE 4. Comparison of base64 encoding and decoding.

2) HOST-BASED FEATURES

Features F39 to F41 are the communication protocol of the connection, information about the connection, and the version of the connection client, respectively. Features F42 and F43 are related to login information. Features F44 to F46 are the duration, average string length of the response, and the presence or absence of a file during the connection. The authors proposed one feature, F45, in the file type.

F45 Received_Size (AVG): The attacker will always query the content through the Linux command. For example, the user types ''uname'' returning the string ''Linux''. The size result is six. The act of stealing information is determined by the length of the string returned. Feature 45 is evaluated using Eq1.

$$ReceivedSize(AVG) = \frac{TotalReceivedSize}{NumberofCommands} \tag{1}$$

Table 5 presents an example of the command calculation. Table 6 presents the details of the host-based features.

3) GEOGRAPHY-BASED FEATURES

Features F47 to F50 are, city names that were analyzed globally. Latitude and longitude are used to determine the location of an attack. The corresponding geographic location can be used to determine whether an abnormal attack has occurred. Table 7 presents the details of the geography-based features.

E. MODEL TRAINING

The proposed AI@NTDS system is designed using the LightGBM algorithm. XGBoost and LightGBM are based on the Tree Boosting mechanism [29]. The LightGBM algorithm is well-known for its better training efficiency and lower memory usage than the XGBoost algorithm. The LightGBM algorithm differs from the traditional Gradient Boosting Decision Tree (GBDT) algorithm and is optimized using various strategies. It has the following four main characteristics; a histogram algorithm, Gradient-based One-Side Sampling (GOSS), Exclusive Feature Bundling (EFB), and Leaf-Wise Tree Growth. The GOSS algorithm is described below Algorithm 1.

When the feature data have many dimensions, other GBDT algorithms have the greatest drawbacks like poor efficiency

TABLE 4. Message-based features.

ID	Name	Description
F1	*Keyword_bash*	Calculate how many " bash " keywords are in the message.
F2	*Keyword_shell*	Calculate how many " shell " keywords are in the message.
F3	*Keyword_exit*	Calculate how many " exit " keywords are in the message.
F4	*Keyword_help*	Calculate how many " help " keywords are in the message.
F5	*Keyword_passwd [user] (Set_account)*	Calculate how many " passwd [user]" keywords are in the message.
F6	*Keyword_chpasswd (Set_account)*	Calculate how many " chpasswd " keywords are in the message.
F7	*Keyword_useradd -[suffix] (Set_account)*	Calculate how many " useradd -[suffix] " keywords are in the message.
F8	*Keyword_\ [file] (Execution_file)*	Calculate how many "\. [file]" keywords are in the message.
F9	*Keyword_sh [file] (Execution_file)*	Calculate how many " sh [file] " keywords are in the message.
F10	*Keyword_./ [file] (Execution_file)*	Calculate how many "./ [file] " keywords are in the message.
F11	*Keyword_perl [file] (Execution_file)*	Calculate how many " perl [file] " keywords are in the message.
F12	*Keyword_python [file] (Execution_file)*	Calculate how many " python [file] " keywords are in the message.
F13	*Keyword_/bin/ [file] (Execution_file)*	Calculate how many " /bin/ [file] " keywords are in the message.
F14	*Keyword_chmod .[file] (Set_permissions)*	Calculate how many " chmod . [file] " keywords are in the message.
F15	*Keyword_sudo su (Set _permissions)*	Calculate how many " sudo su " keywords are in the message.
F16	*Keyword_rm [file] (Delect_record)*	Calculate how many " rm [file] " keywords are in the message.
F17	*Keyword_history -[suffix] (Delect_record)*	Calculate how many " history -[suffix] " keywords are in the message.
F18	*Keyword_cat /etc (Get_sys_info)*	Calculate how many " cat /etc " keywords are in the message.
F19	*Keyword_uname (Get_sys_info)*	Calculate how many " uname " keywords are in the message.
F20	*Keyword_wc (Get_sys_info)*	Calculate how many " wc " keywords are in the message.
F21	*Keyword_crontab (Get_sys_info)*	Calculate how many " crontab " keywords are in the message.
F22	*Keyword_w (Get_sys_info)*	Calculate how many " w " keywords are in the message.
F23	*Keyword_ps (Get_sys_info)*	Calculate how many " ps " keywords are in the message.
F24	*Keyword_free (Get_sys_info)*	Calculate how many " free " keywords are in the message.
F25	*Keyword_lscpu (Get_sys_info)*	Calculate how many " lscpu " keywords are in the message.
F26	*Keyword_nproc (Get_sys_info)*	Calculate how many " nproc " keywords are in the message.
F27	*Keyword_uptime (Get_sys_info)*	Calculate how many " uptime " keywords are in the message.
F28	*Keyword_wget (Network_connect)*	Calculate how many " wget " keywords are in the message.
F29	*Keyword_tftp (Network_connect)*	Calculate how many " tftp " keywords are in the message.
F30	*Keyword_scp (Network_connect)*	Calculate how many " scp " keywords are in the message.
F31	*Keyword_ping (Network_connect)*	Calculate how many " ping " keywords are in the message.
F32	*Keyword_kill (Shutdown_action)*	Calculate how many " kill " keywords are in the message.
F33	*Keyword_reboot (Shutdown_action)*	Calculate how many " reboot " keywords are in the message.
F34	*Count_base64*	Calculate the number of times the message has been in base64.
F35	*Count_Hex*	Calculate the number of times the message has been in hexadecimal.
F36	*Count_url*	Calculate how many URLs are in the message.
F37	*Message_length*	Calculate the total length of the message.
F38	*Messages / sec*	Calculate the length of Message input per second.

TABLE 5. An example of feature F45.

Action	Size
CMD: top	35
CMD: uname	6
CMD: crontab -l	33
Total:	74
Received_Size (AVG):	**24.66**

TABLE 6. Host-based features.

ID	Name	Description
F39	*Protocol*	Which communication protocol is it? If it is Telnet, it is 0, and SSH is 1.
F40	*Src_Port*	The port of the source device is connected with the Honeypot
F41	*SSH_Client_Version*	The time from the connection until the user leaves the Honeypot
F42	*Username*	The user's name in the Honeypot.
F43	*Password*	The user's password in the Honeypot.
F44	*Duration*	Time the user stays in the Honeypot.
F45	*Received_Size (AVG)*	The length of the data returned after entering the command and number of discover commands in the Honeypot.
F46	*File*	All files of the Honeypot collect.

TABLE 7. Geography-based features.

ID	Name	Description
F47	*Continent_Code*	Codes for global continents.
F48	*Country_Name*	The name of a unique territorial subject or political entity.
F49	*Region_Name*	The name of a part of a country.
F50	*City_Name*	The name of a more densely populated and developed area.
F51	*Longitude*	Longitude is a geographic representation of the east-west position of a point on the Earth's surface.
F52	*Latitude*	Latitude is a geographic representation of the north-south position of a point on the Earth's surface.
F53	*Received_Size (AVG)*	The length of the data returned after entering the command and number of discover commands in the Honeypot.
F54	*File*	All files of the Honeypot collect.

and scalability. In this study, LightGBM is used with GOSS algorithms to solve this problem. GOSS maintains the random sampling on the small gradient and introduces a small gradient constant weight. The asymptotic approximation ratio of GOSS is $O(\frac{1}{n_l^j(d)} + \frac{1}{n_r^j(d)} + \frac{1}{\sqrt{n}})$,and the generalization performance in GOSS is $\epsilon_{gen}^{GOSS}(d) = |V_j(d) + V_*(d)|$. For fearure j, $V_j(d)$ is the variance gain for a given decision tree algorithm.

A sorting strategy can optimize the performance without a graph in many features. EFB algorithms reduce the number of dimensions consists two algorithms, which are Greedy Bunding and Merge Exclusive Features (MEF) algorithms. The MEF algorithms can merge multiple features in a bundle by adding the offset parameter. The EFB algorithm reduces the time complexly of original algorithms. It can reduce from $O(\#data \times \#feature)$ to $O(\#data \times \#bundle)$ if $\#bundle \ll \#feature$ It accelerates the speed without reducing the accuracy.

Algorithm 1 Gradient-Based One-Side Sampling

Input: I: training data, d: iterations
Input: a: sampling ratio of large gradient data
Input: b: sampling ratio of small gradient data
Input: *loss*: loss function, L : weak learner
models ← {}, fact ← $\frac{1-a}{b}$
topN ← a × len(I), randN ← b × len(I)
for i=1 **to** d **do**
    preds ← models.predict(I)
    g ← *loss*(I,preds), w ← {1,1,...}
    sorted ← GetSortedIndices(abs(g))
    topSet ← sorted[1:topN]
    randSet ← RandPick(sorted[topN:len(I)], randN)
    usedSet ← topSet + randSet
    w[randSet] × = fact
    newModel ← L(I[usedSet], - g[usedSet], w[usedSet])
    models.append(newModel)

TABLE 8. The learning parameters of LightGBM algorithm.

Learning Parameter	Value
Boosting	Gbdt
Learning Rate	0.1
Number of estimators	100
Max depth	-1
Number Leaves	31

In LightGBM processing, feature analysis is performed based on the parameter settings that are shown in Table 8.

The concept of GBDT is used to calculate the residuals as a generation decision tree. The most effective learning rate in this work is 0.1. The iterative process revealed that the best number of LightGBM estimators was 100. Since LightGBM grows leaf by leaf based on the tree model, the number of leaves here is set to 31.

IV. PERFORMANCE ANALYSIS

This section concerns the performances of the machine learning (ML) mechanism, feature-based analysis, and AI@NTDS system analysis. Features are analyzed and discussed. The most effective detection model algorithm is identified. The following section provides experimental proof of the result.

A. ANALYSIS OF ML MECHANISM

Tables 2, 3, and 8 presents the used dataset, features, and learning parameters, respectively. The authors evaluate the AI prediction model with different multi-classification algorithms to assign malicious payloads to three levels.

Table 9 provides various evaluation indexes and the operation time of each machine learning mechanism.

Many machine learning classification algorithms in the security domain use SVM and Random Forest [30], [31]. In previous works, a Decision Tree and naive Bayes algorithm are used to detect this issue [14]. XGBoost and LightGBM are also well-known classification algorithms. In this work, each algorithm with default parameters are compared in terms of accuracy, precision, recall, F1-score, and computation time metrics.

TABLE 9. Performance of different ML mechanisms.

Algorithms	Accuracy	Precision	Recall	F1-Score	Computation Time
Naive Bayes	11.83%	99.11%	67.72%	71.15%	2.07s
SVM	26.67%	93.53%	67.60%	76.65%	8.86s
Decision Tree	92.51%	98.10%	97.91%	97.99%	1.42s
Random Forest	98.62%	99.76%	99.67%	99.72%	57.83s
XGBoost	98.72%	99.72%	99.72%	99.72%	44.25s
LightGBM	98.72%	98.78%	98.68%	99.73%	5.51s

TABLE 10. Feature analysis with 4 studies.

Testing Features	Accuracy	Precision	Recall	F1_score	Log loss	AUC
Message-based Features	98.19%	99.69%	99.69%	98.18%	3.3474	98.20%
Host-based Features	90.07%	97.07%	97.74%	97.39%	4.8581	83.50%
Geography-based Features	83.82%	91.44%	99.89%	95.35%	3.4473	50.57%
All Features	99.20%	99.75%	99.85%	99.80%	2.9773	98.53%

Following a comprehensive evaluation, the LightGBM was used as the proposed AI@NTDS learning model because the values of any indicator in this study measured were better than average. It required the least computation time, making it easier to deploy in various devices for real-time detection.

Naive Bayes and SVM perform poorly. Although these two algorithms are widely used, they are not suitable for the detection of malicious shell commands in this study.

B. ANALYSIS OF FEATURE-BASED

The features of the AI@NTDS system are divided into host-based, message-based, and geography-based groups. One of the purposes of this group is to find the classification model and identify the essential features. Four case studies were performed, and the results of the relevant analysis are provided in Table 10. In Case 1, message-based features alone resulted in good performance of accuracy and precision. An attacker may attack a target by such means as causing confusion, recon, and deletion. These actions cause the attacker's input character to be more than a low-level threat. In Case 2, only host-based features are used. This study contributed significantly to the returned strings and SSH-related information. Case 3 identified the essential features by observing the distribution of attackers based on geographical features. Latitude and longitude were the critical features, but the accuracy of the model using these features and other indicators combined with all of the features were not as high as in the preceding two case studies The results show that risks and hazards cannot be assessed using only the features that are associated with geographical location. In Case 4, all of the features were used, and the best values of all indicators were obtained.

Finally, 52 dimensions are used to analyze the three types of features to obtain the best model of the detecting system, based on the feature engineering with gradient boosting machine, as shown in Figure 5. The message-based features account for about 50% of the ten features; these are Message_length (F37) and \.\w*(F5) features. The host-based features account for 40% of the top ten features; these are Received_size (F45) and duration (F41) features.

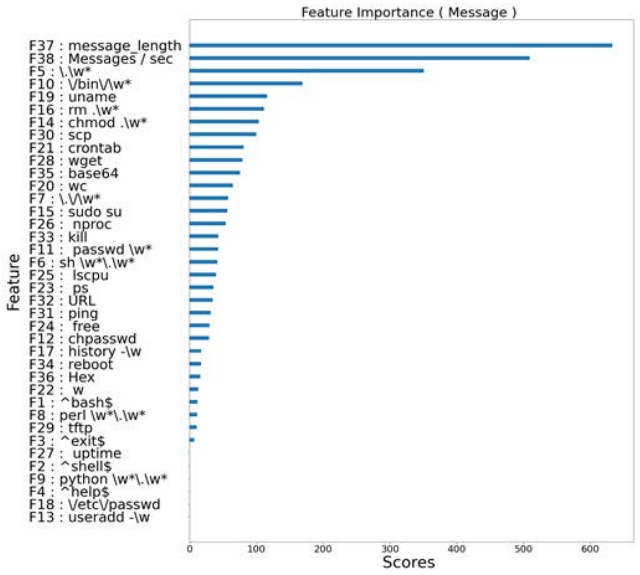
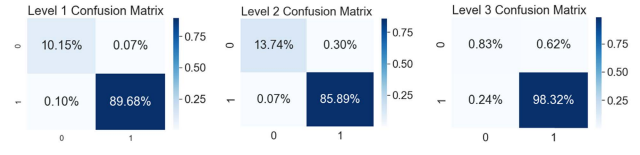


FIGURE 5. Feature importance analysis.

TABLE 11. Performance of AI@NTDS system.



Level 1 confusion matrix	Level 2 confusion matrix	Level 3 confusion matrix

A 99.75% precision, 99.85% recall, and F1-score of 99.80% are achieved. Therefore, the evaluation of AI@NTDS model is based mainly on host-based and message-based features. The proposed features are proved to be very effective.

C. ANALYSIS OF AI@NTDS SYSTEM

The test dataset comprises 23% of the data in all experiment datasets. The training set comprises data from 2019, and the test set consists of data from 2020. The AI@NTDS classifier predicts the classification of each threat in the test dataset, yielding the results in Table 11. From the confusion matrix, the total misclassification ratio of the classifier for threat level 1 is 0.17%; that for threat level 2 is 0.37%, and that for threat level 3 is 0.86%. The F1-score reaches 99.80%, indicating that the AI@NTDS effectively detected samples of various threats. The AUC (Area Under the Curve) reaches 98.53%; the precision rate can reaches 99.75%, and the recall rate reaches 99.85%. Therefore, the detection model that is trained using the LightGBM algorithm can detect malicious sample changes in various periods of attack and has excellent efficiency and performance.

D. COMPARISON WITH OTHER STUDIES

Table 12 compares the performance of the proposed AI@NTDS classifier with those of related methods and algorithms in previous studies. Since previous studies have not provided detailed parameter settings and features of each

TABLE 12. Comparison with different studies.

Title	Our proposed system	Identification and classification of cyber threats through ssh honeypot systems (2020)	Detection of Malicious Remote Shell Sessions (2019)
Dataset	Zenodo Dataset		
Purpose	The study is classified according to different threat levels.		Only benign and malicious.
Algorithm	**LightGBM**	Random Forest	K-NN
Accuracy	**99.20%**	95.664%	94.08%
F1 Score	**99.80%**	98.76%	98.72%
Implementation	Difficult	Medium	**Simple**
Dimension	Medium	**Small**	Large

TABLE 13. Comparison with related works.

Goal	Proposed AI@NTDS System	Classification of SSH Attacks Using Machine Learning Algorithms (2016)	Attack detection and forensics using honeypot in iot environment (2019)	Detection of Malicious Remote Shell Sessions (2019)	Identification and classification of cyber threats through ssh honeypot systems (2020)	A novel Machine Learning-based approach for the detection of SSH botnet infection (2021)
	Classification by level	Classification by malicious and benign	Classification by attack type	Classification by malicious and benign	Classification by level	Classification by infected and uninfected
Message	✓			✓	✓	✓
Host	✓		✓		✓	
Network		✓				✓
Geography	✓		✓		✓	

mechanism, the parameter settings of Random Forest and K-NN were those used in their closest methods when they were originally developed. These mechanisms were compared using the same dataset. The accuracy of the AI@NTDS model is 4% higher than those Random Forest and K-NN, and the F1-score is 1% better. Therefore, the AI@NTDS classifier with the LightGBM algorithm is the most effective in classifying threat levels and identifying the attacker's intent. The difficulties of implementation and the number of data dimensions must be addressed are also compared. Although the model herein yielded better results than both of the other, it requires more features to be extracted from the dataset. Therefore, the proposed mechanism requires more time to spend in preprocessing data. All of the experiment datasets used in Table 12 use the Zenodo dataset, based on the Cowrie Honeypots. The proposed AI@NTDS in this study can be applied to any real-world scenario involves IoT devices and Linux server shell command analysis.

Many studies of related issues have bee performed. Table 13 presents problem-solving mission with various features. The mechanisms that are listed in the first, fourth, and fifth columns have similar purposes and are analyzed in detail Table 12 presents.

V. CONCLUSION

This study proposed an AI@NTDS detection system that incorporates the LightGBM machine learning algorithm for identifying and classifying threats. Attackers' intentions are analyzed using collected data, and the degree of harm that is

caused by malicious instructions is determined. Three types of attack are identified by threat levels of attack are identified using Enterprise Tactics of MITRE ATT&CK. A total of 52 features of three types - message-based, host-based, and geography-based features - are ultimately identified. The results of an analysis demonstrate that our model performed best when all features were used. Message-based features and host-based features accuracy for the model are largest.

REFERENCES

[1] Profit From Tech. *The Ultimate List of Internet of Things Statistics for 2021*. Accessed: Mar. 7, 2021. [Online]. Available: https://www.profitfromtech.com/internet-of-things-statistics/

[2] Venafi. *Secure Shell (SSH) Security, Vulnerabilities and Exploitation*. Accessed: Apr. 19, 2022. [Online]. Available: https://www.venafi.com/education-center/ssh/security-and-vulnerabilities/

[3] Fraunhofer. *Kaiji (Malware Family)*. Accessed: Apr. 19, 2022. [Online]. Available: https://malpedia.caad.fkie.fraunhofer.de/details/elf.kaiji

[4] D. Fraunholz, M. Zimmermann, A. Hafner, and H. D. Schotten, "Data mining in long-term honeypot data," in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 649–656.

[5] A. Kyriakou and N. Sklavos, "Container-based honeypot deployment for the analysis of malicious activity," in *Proc. Global Inf. Infrastruct. Netw. Symp. (GIIS)*, Oct. 2018, pp. 1–4.

[6] S. Kumar, B. Janet, and R. Eswari, "Multi platform honeypot for generation of cyber threat intelligence," in *Proc. 9th IEEE Int. Conf. Adv. Comput.*, Dec. 2019, pp. 25–29.

[7] J. M. Pittman, K. Hoffpauir, and N. Markle, "Primer—A tool for testing honeypot measures of effectiveness," 2020, *arXiv:2011.00582*.

[8] E. Kheirkhah, S. M. P. Amin, H. A. J. Sistani, and H. Acharya, "An experimental study of SSH attacks by using honeypot decoys," *Indian J. Sci. Technol.*, vol. 6, no. 12, pp. 1–12, Dec. 2013.

[9] C. Valli, P. Rabadia, and A. Woodward, "Patterns and patter—An investigation into SSH activity using kippo honeypots," in *Proc. Austral. Digit. Forensics Conf.*, 2013, pp. 1–10.

[10] G. Kambourakis, C. Kolias, and A. Stavrou, "The Mirai botnet and the IoT zombie armies," in *Proc. MILCOM IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2017, pp. 267–272.

[11] C. Kolias, G. Kambourakis, A. Stavrou, and J. Voas, "DDoS in the IoT: Mirai and other botnets," *Computer*, vol. 50, no. 7, pp. 80–84, 2017.

[12] T. Bajtos, P. Sokol, A. Gajdos, K. Lucivjanská, and T. Mézesová, "Analysis of the infection and the injection phases of the TelNet botnets," *J. Universal Comput. Sci.*, vol. 25, no. 11, pp. 1417–1436, 2019.

[13] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: A flow-based SSH intrusion detection system," in *Proc. IFIP Int. Conf. Auto. Infrastruct., Manage. Secur.*, 2012, pp. 86–97.

[14] G. K. Sadasivam, C. Hota, and B. Anand, "Classification of SSH attacks using machine learning algorithms," in *Proc. 6th Int. Conf. IT Converg. Secur. (ICITCS)*, Sep. 2016, pp. 1–6.

[15] T. Bajtoš, P. Sokol, and T. Mézešová, "Virtual honeypots and detection of TelNet botnets," in *Proc. Central Eur. Cybersecur. Conf.*, Nov. 2018, pp. 1–6.

[16] R. M. Arifianto, P. Sukarno, and E. M. Jadied, "An SSH honeypot architecture using port knocking and intrusion detection system," in *Proc. 6th Int. Conf. Inf. Commun. Technol. (ICoICT)*, May 2018, pp. 409–415.

[17] R. K. Shrivastava, B. Bashir, and C. Hota, "Attack detection and forensics using honeypot in IoT environment," in *Proc. Int. Conf. Distrib. Comput. Internet Technol.*, Jan. 2019, pp. 402–409.

[18] P. Dumont, R. Meier, D. Gugelmann, and V. Lenders, "Detection of malicious remote shell sessions," in *Proc. 11th Int. Conf. Cyber Conflict*, May 2019, pp. 1–20.

[19] S. Udhani, A. Withers, and M. Bashir, "Human vs bots: Detecting human attacks in a honeypot environment," in *Proc. 7th Int. Symp. Digit. Forensics Secur. (ISDFS)*, Jun. 2019, pp. 1–6.

[20] T.-H. Lee, L.-H. Chang, and C.-W. Syu, "Deep learning enabled intrusion detection and prevention system over SDN networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, Jun. 2020, pp. 1–6.

[21] J. M. J. Valero, M. G. Pérez, A. H. Celdrán, and G. M. Pérez, "Identification and classification of cyber threats through SSH honeypot systems," in *Handbook of Research on Intrusion Detection Systems*. Hershey, PA, USA: IGI Global, 2020, pp. 105–129.

[22] B. Lingenfelter, I. Vakilinia, and S. Sengupta, "Analyzing variation among IoT botnets using medium interaction honeypots," in *Proc. 10th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2020, pp. 761–767.

[23] J. T. M. Garre, M. G. Pérez, and A. Ruiz-Martínez, "A novel machine learning-based approach for the detection of SSH botnet infection," *Future Gener. Comput. Syst.*, vol. 115, pp. 387–396, Feb. 2021.

[24] B.-X. Wang, J.-L. Chen, and C.-L. Yu, "Cyber security threat intelligence monitoring and classification," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Nov. 2021, pp. 1–3.

[25] A. O. Alzahrani and M. J. F. Alenazi, "Designing a network intrusion detection system based on machine learning for software defined networks," *Future Internet*, vol. 13, no. 5, p. 111, Apr. 2021.

[26] S. Iranmanesh, F. S. Abkenar, A. Jamalipour, and R. Raad, "A heuristic distributed scheme to detect falsification of mobility patterns in internet of vehicles," *IEEE Internet Things J.*, vol. 9, no. 1, pp. 719–727, Jan. 2022.

[27] M. Sewak, S. K. Sahay, and H. Rathore, "Deep reinforcement learning for cybersecurity threat detection and protection: A review," in *Proc. Int. Conf. Secure Knowl. Manage. Artif. Intell. Era*, vol. 1549, 2021, pp. 51–72.

[28] U. Sedlar, M. Kren, J. L. Štefanič, and M. Volk, "CyberLab honeynet dataset (1.0) [data set]," Zenodo, Feb. 29, 2020, doi: 10.5281/zenodo.3687527.

[29] K. Guolin, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T. Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3149–3157.

[30] T. T. T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *IEEE Commun. Surveys Tuts.*, vol. 10, no. 4, pp. 56–76, 4th Quart., 2008.

[31] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

BO-XIANG WANG was born in Taiwan, in 1996. He received the M.S. degree in electrical engineering from the National Taiwan University of Science and Technology, Taipei, Taiwan, in 2021. His main research interests include machine learning, deep learning, and cyber threat intelligence.

JIANN-LIANG CHEN (Senior Member, IEEE) was born in Taiwan, in December 1963. He received the Ph.D. degree in electrical engineering from the National Taiwan University, Taipei, Taiwan, in 1989. Since August 1997, he has been with the Department of Computer Science and Information Engineering, National Dong Hwa University, where he is a Professor and the Vice Dean of the Science and Engineering College. He joins the Department of Electrical Engineering, National Taiwan University of Science and Technology, as a Distinguished Professor. His current research interests include cellular mobility management, cyber security, personal communication systems, and the Internet of Things (IoT).

CHIAO-LIN YU was born in Taipei, Taiwan, in 1997. He received the B.S. degree, in 2020. He is currently pursuing the M.S. degree in electrical engineering with the National Taiwan University of Science and Technology, Taipei. His main research interests include cyber security, vulnerability research, and defense.