Hindawi
Security and Communication Networks
Volume 2020, Article ID 8842539, 14 pages
https://doi.org/10.1155/2020/8842539

Research Article

GroupTracer: Automatic Attacker TTP Profile Extraction and Group Cluster in Internet of Things

Yixin Wu,[1] Cheng Huang,[1] Xing Zhang,[2] and Hongyi Zhou[2]

[1]*College of Cybersecurity, Sichuan University, Chengdu 610065, China*
[2]*NSFOCUS, Beijing 100089, China*

Correspondence should be addressed to Cheng Huang; opcodesec@gmail.com

Received 3 September 2020; Revised 3 November 2020; Accepted 19 November 2020; Published 4 December 2020

Academic Editor: Ting Chen

As Advanced Persistent Threat (APT) becomes increasingly frequent around the world, security experts are starting to look at how to observe, predict, and mitigate the damage from APT attacks. In the meantime, the Internet of things devices are also risky and heavily exposed to the Internet, making them more easily used by hacker organizations to launch APT attacks. An excellent attacker can take down millions of Internet of things devices in a short time. Once the IoT botnet is built, attackers can use it to launch complex attacks which could damage Internet infrastructure and cause network disconnection. This paper proposes GroupTracer, a framework for observing and predicting the Internet of things attacks. GroupTracer is designed to automatically extract the TTP profiles (i.e., tactics, techniques, and procedures) that can describe the behavior of attackers through their tactics, techniques, and processes and dig out the potential attacker groups behind complex attacks. Firstly, it captures attacks by IoT honeypots and extracts relevant fields from logs. Then, attack behaviors are automatically mapped to the ATT&CK framework to achieve automatic TTP profiles extraction. After that, GroupTracer presents four feature groups, including TTP profiles, Time, IP, and URL features, a total of 18 features, mines potential attack groups through hierarchical clustering algorithm, and compares the clustering results with two baseline algorithms. As the ground truth labels are unknown, we apply three internal validation indexes to evaluate the cluster quantity. Experimental results showed that the proposed framework has achieved an excellent performance in exploiting potential groups as the Calinski–Harabasz index reaches 3416.93. Eventually, attack trees are generated for each cluster where nodes indicate attack commands and edges represent command sequences. These attack trees could help better understand each attack group's actions and techniques.

1. Introduction

The Global Research and Analysis Team (GReAT) at Kaspersky points out that the Advanced Persistent Threat (APT) activity has become increasingly complex and destructive [1] since these APT groups launch targeted attacks on critical infrastructure and attempt to compromise central networks. Meanwhile, the Internet of things has become the no. 1 security threat to personal privacy, corporate information security, and even critical infrastructure since IoT devices are inherently risky and easy to exploit while being heavily exposed to the Internet. What is worse, attackers can utilize open-source tools to quickly assemble malware that can scan, penetrate, and control IoT devices. Excellent hackers can take down millions of IoT devices in a short time. Once IoT botnets are formed, attackers can launch an APT attack to hazard the Internet infrastructure and cause network disconnections (e.g., Dyn cyberattack [2] and VPNFilter event [3]). The emerging challenge is how to observe and predict attacks on IoT devices by individuals or even attacker groups since the number of attacks on IoT devices, which are perfect tools for APT attacks, has risen dramatically.

1.1. Describing Individual Behavior. While behavior detection methods for attacks are mostly based on Indicators of Compromise (IOCs) extracted from rule-based methods or traditional blacklists, the information conveyed by such

IOCs is not enough to describe the abundant and varied network security environment due to the following reasons:

(i) IOC is unstable and is easily changed by attackers. For example, if adversaries are leveraging an anonymous proxy service like Tor, they may change IPs quite frequently with little effort and never be noticed.

(ii) IOC cannot express how the attacker interacts with the victim system, and the process of the attack cannot be represented.

(iii) Redundancy occurs when IOC is used to express an attack. In other words, more IOCs do not necessarily lead to a better description.

Bianco proposed the Pyramid of Pain [4], in which each level of the pyramid represents different types of attack indicators leveraged to detect the activities of the adversary, and the most valuable attack indicator is attacker TTPs. TTP profile [5] describes the flow that adversaries go through to accomplish their mission, from initial access to impact and at every step in between, which is abundant to support a comprehensive analysis of the aggressive behaviors of individuals or attack groups. Meanwhile, the defense is shifting from vulnerability-centric to threat-centric, and flexible and efficient security architecture can only be constructed with a sufficient understanding of the threat of the critical assets, which depends on an overall comprehension of the attack tactics, techniques, and behavior patterns (i.e., TTPs). However, at this stage, there is no mature method to normalize the description of attacks on IoT devices and map them to the analysis model. A method for automatic TTP profile extraction of IoT device attacks is expected.

1.2. Clustering Attackers into Groups. With the rapid growth of APT activities, the evolution of a threat landscape moves from a single hacker to well-organized attack actor groups (e.g., Darkhotel [6] and Turla [7]). How to find and depict the behavior of an attack actor group among an ocean of attacks becomes a challenge. Behavioral analysis in sandboxes [8, 9] and binary analysis [10, 11] seem like pleasant ways, which can match malicious samples used by attackers to known or novel malicious families and capture their behaviors to observe the similarities between these attackers. However, malicious family and attack group have a many-to-many relationship, and we cannot just rely on the analysis of malicious samples to find the group behind attacks. Considering the excellent performance of the data-driven approach in the field of network security [12–14], we try to tackle the challenges from a data-driven perspective.

Given the challenges presented above, this paper aims to develop mapping knowledge bases from attacker payloads to the ATT&CK framework to extract the TTP profile and generate behavior fingerprint for attackers to discover groups behind active campaigns. The ultimate purpose is to observe the behavior of attack actor groups and predict attacks in the Internet of things.

1.3. Contributions. Three critical contributions of the paper are as follows:

(i) *Comprehensive Description of Attacker Behavior.* GroupTracer leverages four feature groups (TTP profile, Time, IP, and URL) that are derived from log data to characterize different actions of attackers, which addresses the emerging challenge of the observation and prediction of attacks on IoT devices by individuals. The TTP profile depicts the technique, tactic, and procedure of the attacker. The Time feature group provides statistical characteristics based on attack duration, number of attacks, and time zone of the attacker. The IP and URL feature groups both involve the type of IP/URL and the malicious index, while the latter also analyzes the download file.

(ii) *Automatic TTP Profile Extraction.* Considering that the data source is honeypot log data, which collects payloads utilized by attackers, we construct the 1st and 2nd knowledge bases, which store the mappings between commands and TTPs. By using these knowledge bases, GroupTracer maps commands derived from payloads to the ATT&CK framework to extract the TTP profile, which bridges the gap between cyber threat intelligence (CTI) and the attacker.

(iii) *Group Cluster and Attack Tree Generation.* GroupTracer proposes four feature groups and hierarchical clustering algorithm to build attack group cluster model which aims at finding out the potential groups behind complex attacks. In order to better understand each attack group's behaviors, GroupTracer also introduces attack tree construction method where nodes describe attack commands and edges represent command sequences. The evaluation result shows that GroupTracer can achieve excellent performance as the Calinski–Harabasz index reaches 3416.93.

The remainder of this paper is organized as follows: Section 2 explains the related work about the fundamental techniques used in our framework. Section 3 presents the data collection, flow of feature processing, application of clustering algorithm, and attack tree creation in GroupTracer. The entire experiment and evaluation process is elaborated in Section 4. Finally, the conclusion and future work are discussed in Section 5.

2. Related Work

2.1. Application of IoT Honeypot. To specialize in cyberattacks and defend against them, tools for proactive defense are presented. For instance, honeypot that can capture attacks, document intrusion information about instruments and behaviors of hackers, and prevent attacks outbounding the compromised system [15] has been widely leveraged in cybersecurity. Due to the vulnerable and destructive nature of IoT devices [16–18], the number of

IoT honeypots based on different protocols is rapidly increasing. Currently, some IoT honeypots have already existed [19]. The work in [20] utilizes IoTPOT, a novel honeypot that stimulates the Telnet-enabled IoT devices, which handles commands sent by attack actors, analyzes malicious families on different CPU architectures, and provides an in-depth analysis of ongoing attack behavior. However, IoTPOT focuses on observing the characteristics of malicious families (e.g., spread tendency and ultimate goal) and relationships between these families. It does not employ existing data to analyze the behavior of the aggressors behind attacks and associations between them in detail, which is the center of cyber threat intelligence. A honeypot that emulates the ZigBee gateway and aims at assessing ZigBee attack intelligence and IoT cyberattack behavior is proposed in the text [21]. Although this paper analyzes the commands in the honeypot data at great length and classifies them into six categories of attacks, it does not mine the TTP of these attacks, which helps analysts in threat modeling. Heo and Shin [22] analyze the connection-level log data to study Telnet service scanning to provide solid evidence for the existence of IoT botnet, whereas the dataset contains only connection metadata, so there is no way to analyze the payload in packages, which is a critical evidence for attacking, thus not entirely convincing.

In conclusion, most published methodologies have focused on a single service such as Telnet and ZigBee and analyzed features of malicious families. GroupTracer is more widely used for protocols where command execution vulnerabilities occur and depicts the characteristics of attack behaviors. Besides, most of the previous studies have not analyzed payloads at the TTP level, and some even have not analyzed payloads at all. GroupTracer extracts attack techniques, tactics, and procedures (TTPs) from payloads and utilizes payloads to build attack trees for potential attack groups to more specifically demonstrate their attack behaviors.

2.2. Cyber Threat Intelligence and TTP Extraction. Gartner defines cyber threat intelligence (CTI) as evidence-based knowledge, which can be utilized to inform decisions concerning the subject's response to menace or compromise [23]. With the rapid evolution of the cyber threat landscape, the demand for high quality and fast speed of CTI exchange that allows organizations to respond to emerging threats at the tactical level is becoming increasingly urgent. TTP describes the techniques, tactics, and attack patterns used by the adversary and can be presented in structured text formats that meet the high demand. Husari et al. [24] develop TTPDrill that can achieve the automatic and context-aware analysis of CTI to generate TTPs precisely. Their work bridges the gap between unstructured cyber threat intelligence and structured techniques, tactics, and procedures. Nevertheless, their data source is the cyber threat intelligence, which means that only after CTI is produced, can TTPDrill construct a complete attack pattern. Our work aims at decreasing the time-to-defend even more.

2.3. Group Cluster. Clustering and correlating have been studied extensively and are employed in a multitude of data-driven domains, including security and privacy areas [25]. In a similar direction to this paper, the work in [26] applies an unsupervised method to characterize and classify security-related anomalies and attacks that exist in honeypots without learning phase, labeled traffic, or attack signature database. Cho et al. [27] compare the similarity of the distributed domain to predict the same group, which provides the possibility of response to future attacks. Azevedo et al. correlate IOCs from different OSINT feeds and cluster them to obtain enriched IOCs. This work allows the identification of attacks that was impossible by analyzing IOCs individually. One work that inspires us comes from Ghiëtte et al. [28]. They dissect the SSH protocol to fingerprint tools based on cipher suites and SSH version strings, employing key exchange algorithms and SSH banners to cluster similar tool usage into collaborating individuals and even campaigns. However, as [4] said, adversaries can employ or create another tool that has the same capability to evade detection.

By comparison, GroupTracer employs honeypot log data, in which timestamps, IP addresses, and payloads sent by attackers are usually recorded, and considers four different perspectives (e.g., TTPs and Time) to generate feature groups. For example, it generates TTP profiles by mapping payloads to ATT&CK framework based on command characteristics. Then, it clusters similar adversaries' behaviors into groups based on these features. Our work draws on the strengths of the mentioned studies and improves their weaknesses.

3. Framework

The ultimate goal of this paper is to automatically extract TTP profiles and cluster attack groups in the Internet of things. Figure 1 overviews the flow of GroupTracer. Firstly, it captures attacks, generates raw data, and extracts features from specific fields (e.g., timestamp, payload, and timezone). We deploy numerous honeypots on the Internet to capture attacks. Secondly, it enriches these features. As for generating the TTP profile feature group, it cuts payload into commands, maps these commands to the ATT&CK framework, and then generates Abstract Syntax Tree of the commands for a second mapping to techniques and tactics. After generating all feature groups, encoding and TF-IDF algorithm are utilized to vectorize these string-type features. Thirdly, it combines all feature vectors and leverages the hierarchical clustering algorithm to cluster these attackers into groups. Finally, attack trees for each group, where nodes are commands and edges are command sequences, are created from their payloads to characterize attack profiles.

3.1. Raw Data Collection. The log format of open-source honeypot contains general fields and particular fields. There are 12 general fields standard in all honeypots. src_ip is the source IP, and src_port is the source port number. sensor_ip and dst_port represent the IP and destination port number

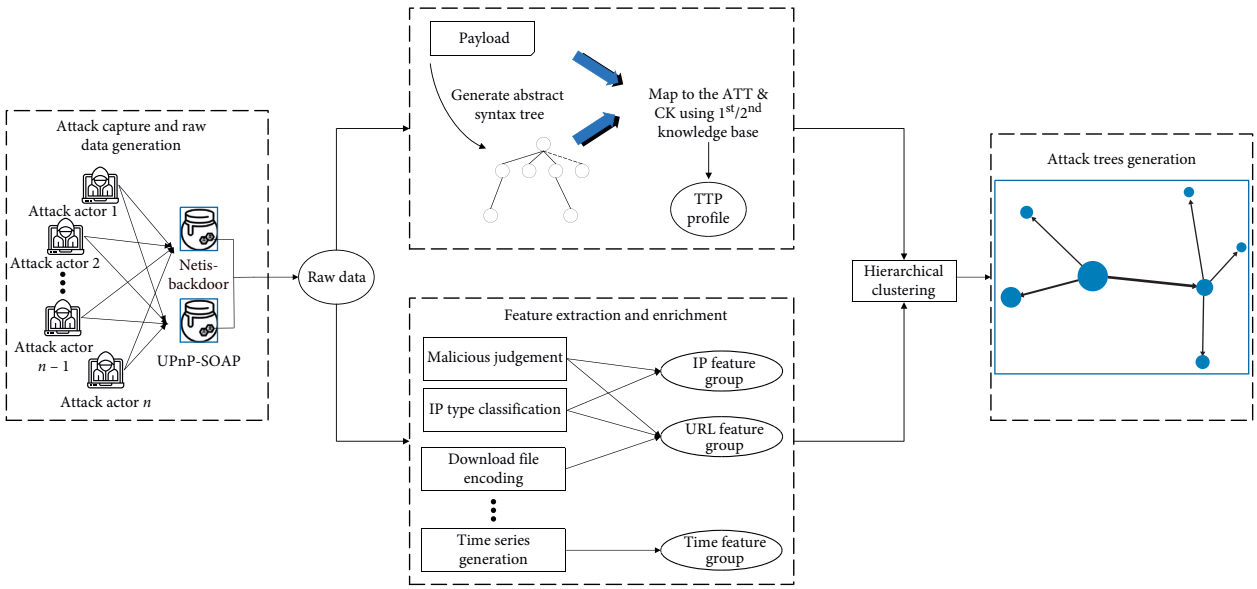

Figure 1: The architecture of GroupTracer.

of the honeypot, respectively. timestamp represents the time when the event occurred. protocol represents the underlying protocol used by the honeypot. geoip describes the current geographic location of the IP and related information like time zone, which is usually obtained by the external API GeoLite [29]. pot_version is added by the honeypot developer to indicate the current version of the honeypot. fingerprint is mostly a hash value generated from a string of src_ip, timestamp, and a specific honeypot-specific field. log _type depicts the type of event recorded in the log, and honeypot_type describes the type of honeypot. container_id represents the ID of the docker container. The particular field is determined according to the specific protocol used by the IoT honeypot. GroupTracer mainly leverages three general fields, namely, src_ip field, *timestamp* field, and geoip field in honeypot log data, as these three fields can provide information to generate the Time and IP feature groups. Given that the payload may appear in different fields, GroupTracer will accurately locate the corresponding field through string matching. To ensure the universality of the framework and the integrity of the payload, the contents of all fields that have payloads are spliced.

3.2. Feature Extraction. The following subsections detail how GroupTracer converts these fields into four feature groups, namely, the TTP profiles, Time, IP, and URL. src_ip field is considered to be the primary key in all fields because the probability of an IP being used by multiple groups is minimal, even if the individual IP is assigned dynamically.

3.2.1. TTP Profile Feature Group Generation. The TTP profile consists of tactics and techniques used by attack actors. Tactic depicts the common strategy of a threat action (e.g., execution and defense evasion). ATT&CK framework provides 12 categories for corresponding techniques. GroupTracer extracted these names as tactics in TTP profiles. Technique describes attack techniques implemented by attack actors under a specific tactic. For example, defense evasion is a tactic that can be performed by a technique named clear command history.

GroupTracer produces the TTP profile primarily from a command execution perspective. There is a crucial issue to be compromised. On the one hand, the classification of commands should be as accurate as possible. On the other hand, command parameter values sometimes affect the classification needlessly. The following two commands illustrate both cases. Both of these commands can be classified as technique file deletion. (i) can be further subdivided into technique clear command history, whereas (ii) can only be classified as technique *file deletion*. For (i), the whole statement is a valid classification basis, while for (ii), only *rm* can serve as a valid classification basis, and all other parts are redundant.

(i) *rm -rf ~/*.bash_history

(ii) *rm -rf xb.sh xb.sh xb2.sh xb1.sh*

Aiming at solving the above dilemma, the quadratic mapping method is proposed. Figure 2 illustrates the process of quadratic mapping. There are two knowledge bases in GroupTracer. One saves the mapping of the entire command statement to tactics and techniques for the first mapping, and the other stores the mapping of the abstract command structure to tactics and techniques for the second mapping. In the first mapping, GroupTracer splits the original payload into several commands and maps these commands to the 1st knowledge base to attain some of the TTP profiles. As shown in Figure 3, GroupTracer generates the Abstract Syntax Tree [30] for each payload to obtain command nodes and extract abstract command structures. Then, these abstract structures are put as input into the 2nd knowledge base to acquire the

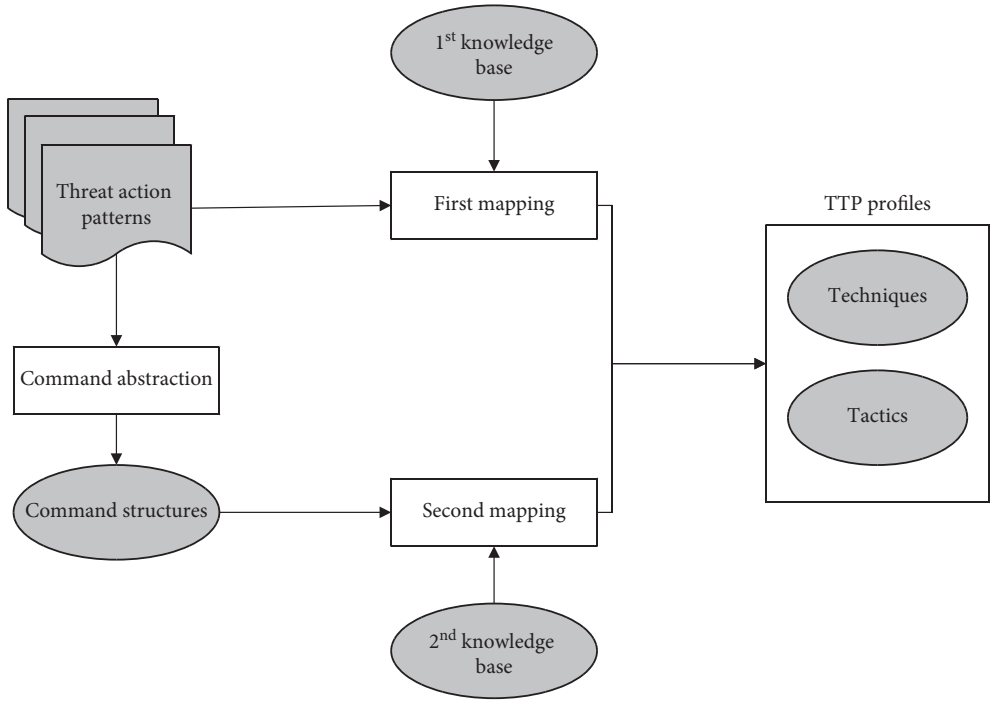

Figure 2: The process of generating the TTP profile. GroupTracer cuts payload into commands, maps these commands to the ATT&CK framework, and then abstracts the structure of the commands for a second mapping to techniques and tactics. The product of the first mapping and the second mapping constitutes the TTP profiles.

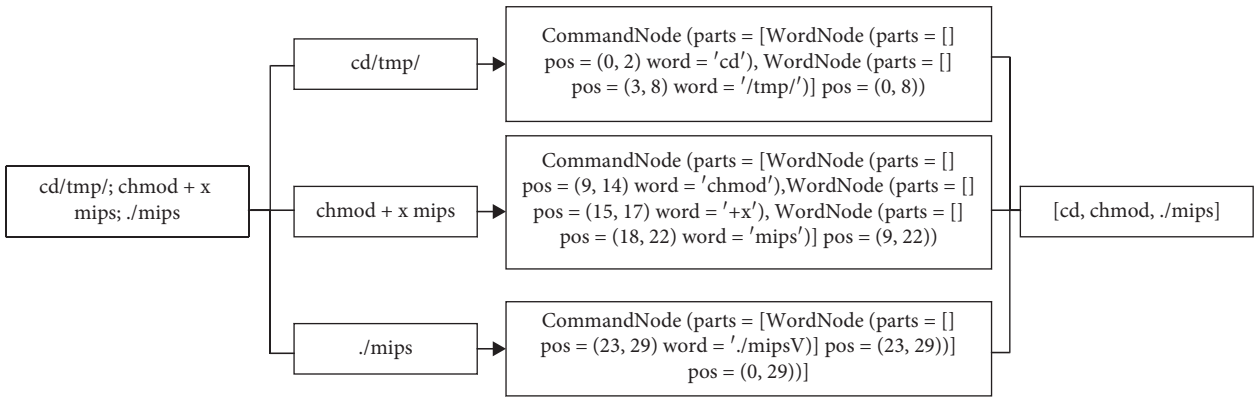

Figure 3: GroupTracer generates the Abstract Syntax Tree for each payload to obtain command nodes and then extracts the abstract structure from command nodes.

remaining TTP profiles. Table 1 only shows some mapping samples due to space limitation. After obtaining the TTP profiles, the GroupTracer encodes the corresponding string into a numeric feature vector. The final data structure can be described as follows:

$$\text{TTP} = [\text{techniques}, \text{tactics}], \quad (1)$$

where there are variable-length techniques and tactics.

3.2.2. IP and URL Feature Groups Generation. The features related to IP and URL feature groups are shown in Table 2, the first three of which are common to both feature groups, and the last one is unique to the URL feature group. The country can be obtained through Ipdb [31]. VirusTotal [32] provides multiple antivirus scanning engines (e.g., Kaspersky URL advisor, Malware Domain Blocklist [33], and Dr.Web Link Scanner [34]) used for URL scanning. GroupTracer first utilizes VirusTotal to scan for unknown IPs/URLs and then regards the number of antivirus engines that return malicious results as the malicious index for those IPs/URLs. According to the purpose, there are seven types of IP addresses shown in Table 3. This framework uses the RTBAsia API [35] to get the classification of IP types. Download is an optional option for the URL feature group, depending on the type of vulnerability in the honeypot, because in some cases the download is to install a backdoor for subsequent manipulation, while in others it is to provide

Table 1: Some examples of mapping from commands to TTP profiles.

Command	Technique	Tactic
1st knowledge base		
show running-config	Credential dumping	Credential access
show startup-config	Credential dumping	Credential access
2nd knowledge base		
ftpget	Remote file copy	Lateral movement
wget	Remote file copy	Lateral movement
curl	Remote file copy	Lateral movement
rcp	Remote file copy	Lateral movement
copy	Remote file copy	Lateral movement
show archive config	Credentials in files	Credential access
show history	Input capture	Collection
show logging	Input capture	Collection
tar	Data compressed	Exfiltration
zip	Data compressed	Exfiltration
rar	Data compressed	Exfiltration
shutdown	System shutdown/reboot	Impact
reboot	System shutdown/reboot	Impact
del	File deletion	Defense evasion
rm	File deletion	Defense evasion
adduser	Create account	Persistence
usermod	Account manipulation	Persistence
groupadd	Account manipulation	Persistence
dir	File and directory discovery	Discovery
ls	File and directory discovery	Discovery
cd	File and directory discovery	Discovery
echo	Data from local system	Collection
cat	Data from local system	Collection
more	Data from local system	Collection
pwd	Data from local system	Collection
whoami	Data from local system	Collection

some tools for privilege escalation under certain circumstances. When attacking different honeypots, it is likely that hackers in a group download file for different purposes, which will greatly affect our performance in clustering only by downloading file names. After attaining these feature groups, the GroupTracer encodes the corresponding string into a numeric feature vector.

3.2.3. Time Feature Group Generation. As the attacker groups tend to use the tool framework to attack the specified target, there is a corresponding regularity in the IP time zone, attack duration, number of attacks, etc. Algorithm 1 describes the generation of the basic time-series features for a given T_{ip}, which represents the set of timestamp for a specific IP. $\widehat{T}$ is a collection of T used as input to generate the start time, the attack duration, and the number of attacks in each duration. In addition, GroupTracer reuses this code to select the final threshold. When selecting threshold, we nest a for loop in the outermost layer, and the threshold value increases by 1. Meanwhile, we count the number of time_interval in each T_{ip} and assign their sum to the variable num_interval. If num_interval has not changed compared to the previous loop, we jump out of the cycle and output the final threshold.

The threshold, which indicates how small the time interval between two attacks is before they are considered to belong to the same attack period, is utilized to divide the attack duration. Partitioning each attack period and characterizing the behavior (e.g., duration and number of attacks) of each attack steadily can be the key to the reliability of time analysis, since the members of a group may be similar in these respects. To select a number as the initial threshold, GroupTracer maps all the time interval values into the following 5 time buckets: <1 s, [1 s, 1 min], (1 min, 1 h], (1 h, 1 day], >1 day. The results show that 99.91% of time gaps fall into the first four buckets, so the initial threshold is set to 1 h. Then, GroupTracer employs Algorithm 1 to accomplish the threshold selection procedure. It first counts the total number of attack durations for each IP and then adjusts the threshold slowly until the number of attack periods for most IPs is almost unchanged. If multiple thresholds have the same result, choose the smaller one first, as the threshold always tends to choose the smaller one.

After we get the final threshold and basic time-series features, we extract the statistical features from the basic ones, namely, the start time, the attack duration, and the number of attacks in each attack duration. GroupTracer draws a new time series for the relevant features of each IP, taking the start time as the independent variable and the attack duration and number of access as the dependent variables. Several features that proved to be significant time-series characteristics in an early stage shown in Table 4 are applied. After generating the time series, GroupTracer will automatically calculate these selected features to produce the final feature vectors.

By encoding the timezone field appearing in the data, GroupTracer finally turns the string-type features into integer vectors. Eventually, statistical features from timestamp and encoded time zone constitute the Time feature group.

3.3. Group Clustering Algorithm. As the hacker groups tend to utilize customized frameworks, features like time gap and TTP profiles have specific patterns. Therefore, attacker behavior naturally forms clusters. This paper is aimed at identifying such natural clusters to dig out potential groups behind active campaigns.

We employ the well-known hierarchical clustering algorithm [36] as it captures the hierarchical structure between clusters, which helps security experts to observe the relationship between clusters and subclusters. Moreover, hierarchical clustering is suitable for arbitrary shape clustering and is insensitive to the input order of samples.

Table 2: Features related to IP & URL feature groups.

#	Feature name	Description
1	Country	Describes the country to which the IP/URL belongs.
2	Malicious index	Leverages the VirusTotal API to determine the maliciousness of the IP/URL.
3	IP address type	Utilizes the RTBAsia API to classify IP/URL type.
4	Download (optional)	The file that the attack actor downloaded by executing the command.

Table 3: Seven types of IP addresses.

#	Types
1	Internet data center
2	Fixed IP Internet access line
3	Ordinary broadband
4	Mobile broadband
5	Backbone node
6	Known crawler API
7	Small operator

Figure 4 depicts a bottom-up approach to perform hierarchical clustering. Each sample represents a unique cluster in the beginning, and then we choose Euclidean distance to calculate the similarity between each cluster and merge these clusters successively. The threshold applies when forming the final flat clusters. The Euclidean distance is computed as follows [37]:

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \quad (2)$$

where $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ are two points in Euclidean n-space.

3.4. Attack Tree Creation Method. After digging out potential groups, GroupTracer gathers all payloads and generates attack trees for each cluster to embody and better understand group behaviors. Algorithm 2 is to process all payloads from a cluster. P_T represents all payloads collected from a given cluster T.

In the directed graph, nodes are command names and edges represent the command sequence between two commands. The out-degree of each command determines the size of each node. If a command exists only at the end of the payload, its size can also depend on the in-degree. The width of each edge is decided by the weight, which describes the frequency of occurrence of the command sequence. For instance, we have some payloads from a cluster, whose command sequences shown in Table 5 are obtained after command segmentation and abstraction. GroupTracer runs Algorithm 2 to generate the attack tree, as illustrated in Figure 5. Line 3–line 13 generate a list that stores two-step command sequences. For example, it turns cd chmod ./t into [cd chmod, chmod ./t]. Line 14–line 16 counts the number of occurrences of all two-step command sequences.

4. Experiment and Metrics

4.1. Dataset. In this section, we first describe the datasets obtained from two kinds of IoT honeypots: UPnP-SOAP multiport honeypot and the Netis router backdoor honeypot. The time of data collection, the number of IPs, and the number of log entries are shown in Table 6.

4.1.1. UPnP-SOAP Multiport Honeypot. In the UPnP service, SOAP protocol assists in defining device types and other related information [38]. Therefore, there are a huge number of IoT devices that provide SOAP services, some of which do not require authentication. Honeypot simulates the behavior of the 11 ports most frequently scanned (e.g., 52881, 5500, and 2048) and the relevant SOAP service path and returns the corresponding information after being scanned. Six nodes are deployed on the Internet, averaging about 700 log entries per day. The dataset contains 153,413 log entries from 2,652 IPs over 196 days in 2019 (Table 6). Each log entry is identified by src_ip, timestamp, timezone, body, and query_string. The src_ip in our datasets is globally unique.

4.1.2. Netis Router Backdoor Honeypot. The Netis router listens on port 53413 (UDP) by default. After sending a specific string to it, the attacker can gain root login and then execute the corresponding command to perform a series of malicious behaviors. The honeypot has seven nodes deployed on the Internet, averaging about 300 log entries per day. Our Netis dataset contains 241,593 log entries from only 373 IPs over 279 days in 2019 (Table 6). Unlike UPnP-SOAP honeypot, the log entry in Netis is characterized by src_ip, timestamp, timezone, and data.

Given that both types of honeypots are simulated command execution vulnerabilities, we can leverage commands executed by attackers to discover their technology and tactics. Therefore, GroupTracer can be utilized to analyze the data of these two honeypots at the same time. Our datasets contain ten types of techniques that can be grouped into six tactics. These tactics are as follows [39]:

(i) Defense evasion: avoiding being detected while adversaries are intruding on the victim system.

(ii) Discovery: figuring out the environment of the victim system.

(iii) Lateral movement: moving through the victim environment. Adversaries might download their tools from remote servers to achieve lateral movement.

Require: $\widehat{T}$
Ensure: Basic Time-series features for all IPs
threshold = t_0
for all $T_{ip} \in \widehat{T}$ **do**
  sort By Time (T_{ip})
  start = end = $T_{ip}[0]$
  for $i \in [0, \text{len}(T_{ip}) - 1]$ **do**
    delta = $T_{ip}[i+1] - T_{ip}[i]$
    if delta ≤ threshold **then**
      end = $T_{ip}[i+1]$
      cnt = cnt + 1
    else
      unit = {start′: start, ′duration′: start − end, ′cnt′: cnt}
      time_interval.append (unit)
      start = $T_{ip}[i+1]$
      end = start
      cnt = 1
    end if
  end for
  if start ≠ end **then**
    unit = {start′: start, ′duration′: start − end, ′cnt′: cnt}
    time_interval.append (unit)
  end if
  time_intervals.append (time_interval)
end for

Algorithm 1: Basic time-series feature generation for a given T_{ip}.

Table 4: Several significant time-series characteristics.

#	Feature name	Description
1	maximum	The largest value of the time series
2	minimum	The smallest value of the time series
3	length	Number of attack periods per IP
4	mean	A measure of the central tendency
5	median	The "middle" value
6	standard_deviation	The square root of its variance
7	variance	The expectation of the squared deviation of a random variable from its mean
8	sum_value	Calculates the sum over the time-series values

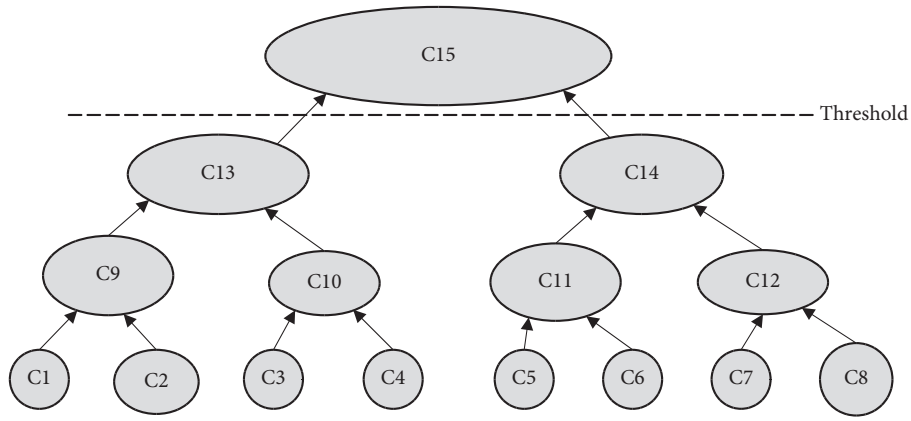

Figure 4: The hierarchy of the attacker behavioral clusters.

```
Require: Payloads P_T
Ensure: Attack tree for T
edges = []
weighted_edges = {}
for all payload ∈ P_T do
    payload.ast()
    tmp = payload.split()
    if len(tmp) > 2 then
        for i ∈ [0: len(tmp) − 1] do
            edges.append(tmp[i: i + 2])
        end for
    else
        edges.append(tmp)
    end if
end for
for all edge ∈ edges do
    weighted_edges[edge] = weighted_edges.get(edge, 0) + 1
end for
G = Digraph()
G.add_edges(weighted_edges)
DrawAttackTree(G, width = weighted_edge, node_size = G.degree())
```

Algorithm 2: Attack tree generation for a given cluster T.

Table 5: The occurrence frequency statistics of command sequence from a certain cluster.

#	Command sequence	Frequency
1	cd wget	5
2	cd rm	2
3	cd chmod./t	2
4	cd chmod./s	1
5	cd chmod./nig	1

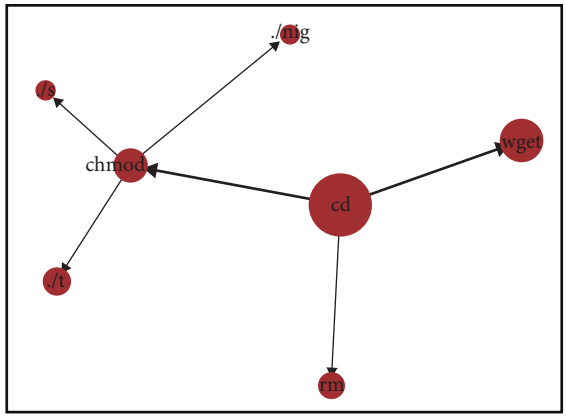

Figure 5: The attack tree for a given cluster.

(iv) Execution: running malicious code. The purpose of adversary-controlled code might be communicating with C&C servers or stealing data.

(v) Impact: expanding the impact of the intrusion on the victim system.

Table 6: Datasets from UPnP-SOAP and Netis backdoor honeypots.

Dataset	Time	# of unique sources	# of entries
UPnP-SOAP	Apr.21–Nov.03 2019	2,652	153,413
Netis backdoor	Mar.21–Dec.25 2019	373	241,593
Total	Mar.21–Dec.24 2019	3,025	395,006

(vi) Collection: gathering information related to the adversary's objectives.

Table 7 shows techniques corresponding to each tactic and commands mapped to these techniques in our datasets.

4.2. Clustering Performance Evaluation. After clustering, we need measurement indicators to evaluate the effect. In general, the measurement for the quality of a clustering algorithm can be categorized into two kinds of criteria [40]: internal validation and external validation. External criteria are based on the previous knowledge about the data and require that ground truth labels are known. However, the labels of samples in this paper are not available. Thus, the internal validation is more suitable for our evaluation. More specifically, three internal indexes are utilized in this evaluation. These indexes measure if clusters are well compact and separated.

(i) Calinski–Harabasz (CH) [41]:

The index CH is defined as follows:

Table 7: Techniques corresponding to each tactic and commands mapped to these techniques in our datasets.

Tactic	Technique	Command
Defense evasion	Disabling security tools	service iptables stop
	Clearing command history	history -c
	File deletion	rm
	File and directory permissions modification	chmod
Discovery	Process discovery	ps
	File and directory discovery	cd; ls; dir
Lateral movement	Remote File copy	tftp; wget; curl; ftpget
Execution	Exploitation for client Execution	sh;./mips;./zuki;./nig
Impact	Network denial of service	ultimate
Collection	Data from local system	echo; more; cat

Table 8: The evaluation of GroupTracer and two comparison baselines using the Calinski–Harabasz, the Silhouette Coefficient, and the Davies–Bouldin index.

Algorithm	Calinski–Harabasz	Silhouette Coefficient	Davies–Bouldin	# of clusters	Value
GroupTracer	3416.9311	0.5389	0.7367	4	10 (threshold)
K-means model	1212.7809	0.3246	1.1807	3	3 (K)
Meanshift model	2199.9206	0.5004	0.6769	2	50% (quantile)

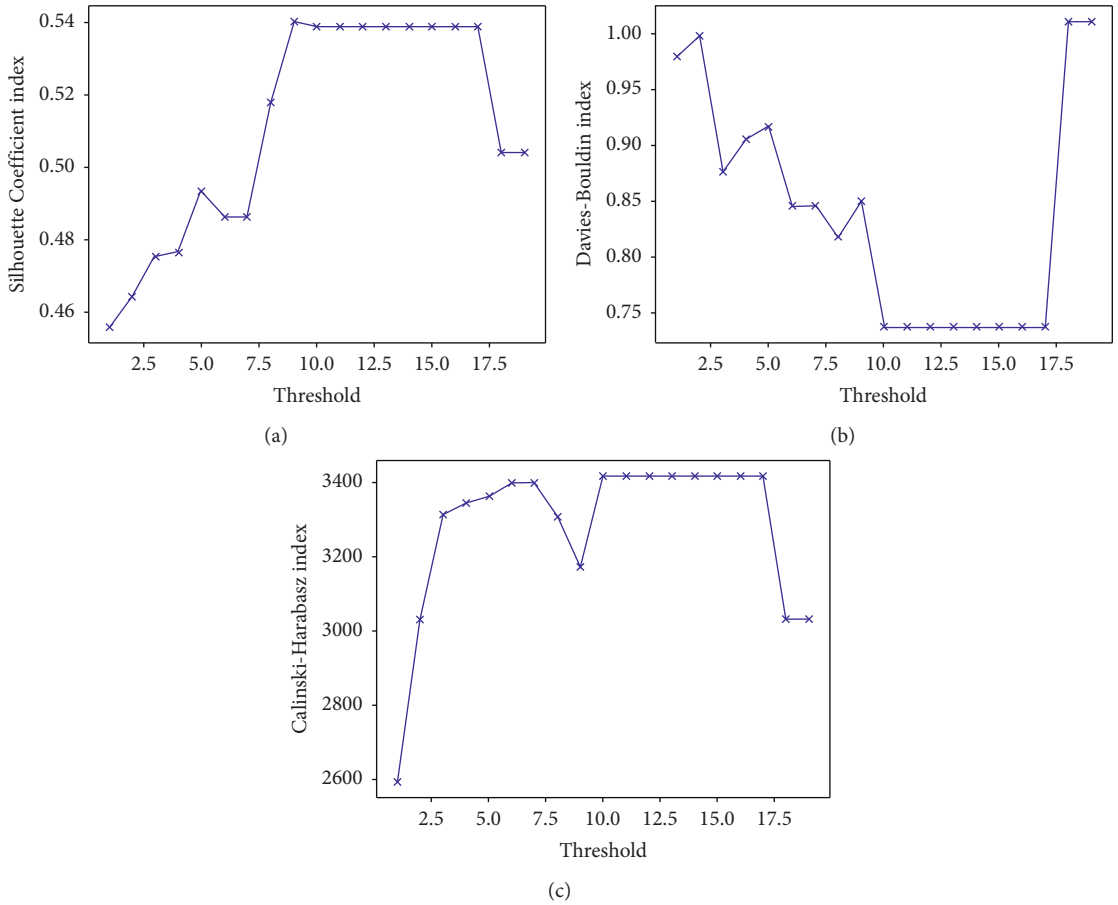

Figure 6: The evaluation of GroupTracer using the three indicators, where the value of threshold ranges from 1 to 20: (a) Silhouette Coefficient; (b) Davies–Bouldin; (c) Calinski–Harabasz.

$$\mathrm{CH} = \left(\frac{\mathrm{trace}(S_B)}{\mathrm{trace}(S_W)}\right) \cdot \left(\frac{n_p - 1}{n_p - k}\right), \tag{3}$$

where S_B denotes the between-cluster scatter matrix and S_W is the within-cluster scatter matrix. n_p is the number of samples, and k represents the number of

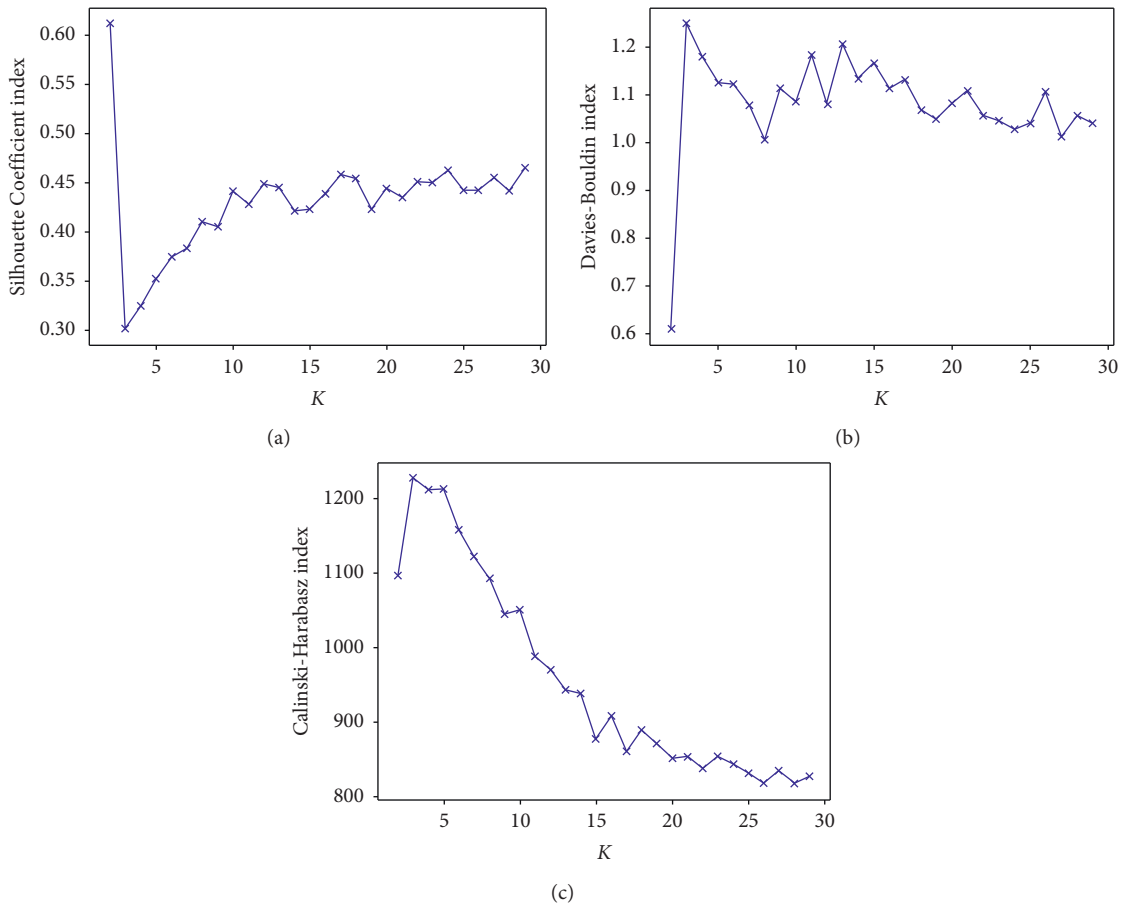

Figure 7: The evaluation of K-means model using the three metrics: (a) Silhouette Coefficient; (b) Davies–Bouldin; (c) Calinski–Harabasz.

classes. The larger value of CH indicates a better clustering solution.

(ii) Silhouette Coefficient [42]:

The Silhouette Coefficient s is composed of two scores: a means distance between a sample and the rest in the same cluster. b is the distance between a point and all other samples included in the next nearest class. s for all samples is given as the mean of the Silhouette Coefficient for each point. As for single sample i, s can be computed by

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}. \tag{4}$$

The Silhouette Coefficient index ranges from −1 to 1; −1 represents a weak clustering effect, and 1 means a good classification effect. 0 indicates the overlap of clusters. A higher s indicates a better clustering quality.

(iii) Davies–Bouldin (DB) [43]:

DB can be measured as follows:

$$\mathrm{DB} = \left(\frac{1}{k}\right) \sum_{i=1}^{k} \max_{i \neq j} \left\{ \frac{d(X_i) + d(X_j)}{d(k_i, k_j)} \right\}, \tag{5}$$

k denotes the number of clusters. i, j represent different cluster labels ($j \neq i$). $d(X_i)$ and $d(X_j)$ are the distance from all samples in cluster i and j to their respective cluster centroids. $d(k_i, k_j)$ is the distance between these centroids. A smaller DB value means a better clustering result.

4.3. Experiment Design. In the following, we evaluate GroupTracer by examining the cluster quality, i.e., how well clusters capture similar attack actors. The primary evaluation is for the group clustering of GroupTracer. We compare the evaluation results of GroupTracer based on the three indicators we mentioned above with the performance of group clustering based on the other two baseline algorithms to conclude that GroupTracer has excellent performance.

4.3.1. Comparison Baselines. Meanshift [44] and K-means [45] clustering algorithms are chosen to be the baselines. Meanshift is a centroid-based algorithm that requires an iterative step. It continuously calculates the expected moving distance of the center point and moves until the final condition is reached. For a given sample set, K-means divides it into K clusters, minimizing a criterion (e.g., within-cluster sum-of-squares). K is a positive integer number and must be predefined.

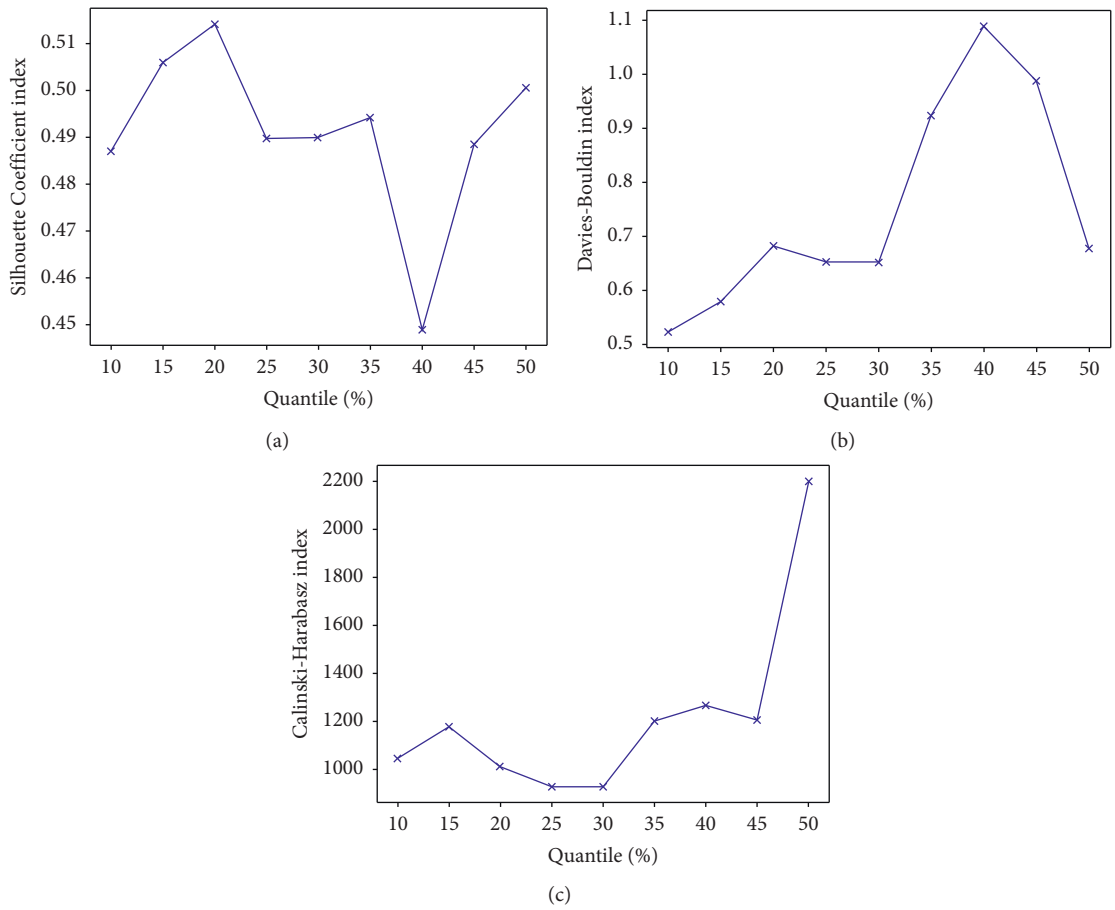

Figure 8: The evaluation of Meanshift model using the three metrics: (a) Silhouette Coefficient; (b) Davies–Bouldin; (c) Calinski–Harabasz.

We first extract the required features according to Section 3.2 and convert them into usable feature matrices. Before performing the final group clustering, we normalize the feature matrices. The reason is that clustering algorithms all use a distance measure to determine if object i is more likely to belong to the same cluster as object j than the same cluster as object k. These distance measures are affected by the scale of the variables. By putting all variables into the same range, we can weigh all variables equally, especially when the feature vectors are generated differently.

The main idea of normalization is to calculate the $p-$ norm of each sample and then divide each element in the sample by the norm. The result of this process is that the $p-$ norm of each processed sample is equal to 1. In the experiment, p is set to 2 and $L2-$ norm of vector $x = (x_1, \ldots, x_n)$ can be defined as [37]

$$\|x\|_2 = \left(|x_1|^2 + |x_2|^2 + \cdots + |x_n|^2\right)^{1/2} = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}. \quad (6)$$

For the purpose of eliminating the influence of irrelevant variables as much as possible, we leverage all data in the dataset for experiments of each algorithm. For GroupTracer, we iterate the threshold value to pick the threshold with the highest clustering quality. The number of iterations is 20. Then we generate multiple versions of K-means clustering to attain the best clustering solution (Calinski–Harabasz index). Meantime, we run the Meanshift algorithm with the window size changing to get the best effect.

4.4. Experiment Result and Discussion. The evaluations of GroupTracer and two comparison baselines using the metrics we mentioned above are shown in Table 8. When running each algorithm, the value of different independent variables (e.g., threshold and quantile) brings the most significant change to the Calinski–Harabasz index, so this index is considered as the primary reference index when evaluating each algorithm. As a result, the Calinski–Harabasz index of GroupTracer reaches 3416.93 when the threshold is set to 10, which is about three times the K-means model and 1.5 times the Meanshift model. Taking the Silhouette Coefficient index as the definitive reference, the performance of GroupTracer is still the best. Although our algorithm performance is not the best after taking the Davies–Bouldin index into account, the gap is also within the acceptable range. GroupTracer generates 4 clusters, while the K-means model generates three classes. In the same way, the Meanshift model only generates 2 clusters for all data.

The evaluation of GroupTracer using the three metrics is illustrated in Figure 6, where the value of threshold ranges from 1 to 20. The Silhouette Coefficient index is on the rise

until the threshold reaches 9. When the threshold value ranges from 10 to 17, the index stabilizes at a relatively high level in Figure 6(a). The Davies–Bouldin index shows a downward trend as a whole until the threshold reaches ten and starts to stabilize at the lowest point (0.7367). After that, the index starts to rise rapidly in Figure 6(b). When the threshold value is 10–17, the Calinski–Harabasz index is maintained at the highest level (3416.93) in Figure 6(c), and the number of clusters is also kept at 4. In summary, all three indicators show that when the number of clusters is 4, the cluster quality becomes the highest and reaches a stable state.

When the CH index reaches the highest point, the Silhouette Coefficient and the DB index both have the worst effect in Figure 7. Similarly, when the CH index reaches the highest level, the DB index is in a lower position in Figure 8. It can be seen in these two figures that the changing trends of these indicators are not matched, which means that the two baselines are not well applied to our datasets.

5. Limitation and Future Work

Our research proposes a framework that can dig out potential groups behind active campaigns. This new technique can make full use of information from attack campaigns. However, our current design is preliminary. We only focus on the IPs used by the potential groups and do not go any further to track which specific groups were involved, that is, to try to correspond to the real groups [46]. Moreover, GroupTracer can only deal with honeypot logs that contain attack payloads.

In future work, we expect to combine NLP techniques with cyber threat intelligence to precisely match these potential groups to the real-world APT groups, in which the attack tree may be helpful to extract a group profile. Moreover, an experiment to prove the effectiveness of the attack tree should also be carried out. Further, we expect to apply more data sources such as system log and network traffic to expand our knowledge base and perform more comprehensive analysis.

6. Conclusion

In this work, we propose GroupTracer, a framework for attack actors clustering from IoT honeypot logs. GroupTracer is aimed at extracting the TTP profile automatically and digging out potential groups behind active campaigns. By mapping payloads to the ATT&CK framework, GroupTracer can effectively extract structured TTP profiles using two knowledge bases. Besides, this framework leverages four feature groups (namely, Time, TTPs, IP, and URL), a total of 18 characteristics, derived from log entries to capture the natural hierarchical structures for attacker groups. Finally, GroupTracer constructs attack trees for each cluster to embody the group actions. In the experiment, we compare our algorithm with two baseline algorithms. The evaluation of 395,006 log entries from 3,025 IPs reveals the high performance of GroupTracer, in which the Calinski–Harabasz index reaches 3416.93. Moreover, our proposed framework is generalizable as it is from a log accounting perspective, so its application is not limited to the IoT honeypot.

Data Availability

The research data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by National Natural Science Foundation of China (61902265) and National Key Research and Development Program (2016YFE0206700 and 2018YFB0804503).

References

[1] GReAT, Apt Trends Report, 2019, https://securelist.com/apt-trends-report-q2-2019/91897/.

[2] H. Scott, "Dyn," 2019, https://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/.

[3] Talos, Vpnfilter, 2019, https://blog.talosintelligence.com/2018/05/VPNFilter.html.

[4] D. J. Bianco, "The pyramid of pain," 2019, https://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html.

[5] J. Friedman and M. Bouchard, *Definitive Guide to Cyber Threat Intelligence: Using Knowledge about Adversaries to Win the War against Targeted Attacks*, Isightpartners, Amsterdam, Netherlands, 2015.

[6] MITRE, Darkhotel, 2019, https://attack.mitre.org/groups/G0012/.

[7] MITRE, 2019, Turla, https://attack.mitre.org/groups/G0010/.

[8] M. Alazab, "Profiling and classifying the behavior of malicious codes," *Journal of Systems and Software*, vol. 100, pp. 91–102, 2015.

[9] S. S. Hansen, T. M. T. Larsen, M. Stevanovic, and J. M. Pedersen, "An approach for detection and family classification of malware based on behavioral analysis," in *Proceedings of the 2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–5, IEEE, Kauai, HI, USA, February 2016.

[10] M. Ahmadi, D. Ulyanov, S. Semenov, M. Trofimov, and G. Giacinto, "Novel feature extraction, selection and fusion for effective malware family classification," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pp. 183–194, ACM, New Orleans, LA, USA, March 2016.

[11] A. Makandar and A. Patrot, "Malware analysis and classification using artificial neural network," in *Proceedings of the 2015 International Conference on Trends in Automation, Communications and Computing Technology (I-TACT-15)*, pp. 1–6, IEEE, Bangalore, India, December 2015.

[12] K. Borgolte, C. Kruegel, and G. Vigna, "Meerkat: detecting website defacements through image-based object recognition," in *Proceedings of the 24th {USENIX}Security Symposium ({USENIX}Security 15)*, pp. 595–610, Washington, DC, USA, August 2015.

[13] T. Taylor, X. Hu, T. Wang et al., "Detecting malicious exploit kits using tree-based similarity searches," in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, pp. 255–266, ACM, New Orleans, LA, USA, March 2016.

[14] Z. Tu, R. Li, Y. Li et al., "Your apps give you away: distinguishing mobile users by their app usage fingerprints," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–23, 2018.

[15] F. Zhang, S. Zhou, Z. Qin, and J. Liu, "Honeypot: a supplemented active defense system for network security," in *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pp. 231–235, IEEE, Chengdu, China, August 2003.

[16] J. A. Jerkins, "Motivating a market or regulatory solution to iot insecurity with the mirai botnet code," in *Proceedings of the 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 1–5, IEEE, Las Vegas, NV, USA, January 2017.

[17] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A. R. Sadeghi, and S. Tarkoma, "Iot sentinel: automated device-type identification for security enforcement in iot," in *Proceedings of the 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, pp. 2177–2184, IEEE, Atlanta, GA, USA, June 2017.

[18] A. K. Simpson, F. Roesner, and T. Kohno, "Securing vulnerable home iot devices with an in-hub security manager," in *Proceedings of the 2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pp. 551–556, IEEE, Seattle, WA, USA, March 2017.

[19] M. Wang, J. Santillan, and F. Kuipers, "Thingpot: an interactive internet-of-things honeypot," 2018, https://arxiv.org/abs/1807.04114.

[20] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, "Iotpot: analysing the rise of iot compromises," in *Proceedings of the 9th {USENIX} Workshop on Offensive Technologies {WOOT}*, Washington, DC, USA, August 2015.

[21] S. Dowling, M. Schukat, and H. Melvin, "A zigbee honeypot to assess iot cyberattack behaviour," in *Proceedings of the 2017 28th Irish Signals and Systems Conference (ISSC)*, pp. 1–6, IEEE, Co Kerry, Ireland, June 2017.

[22] H. Heo and S. Shin, "Who is knocking on the telnet port: a large-scale empirical study of network scanning," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 625–636, ACM, Incheon, South Korea, June 2018.

[23] Gartner, The Definition of Cyber Threat Intelligence, 2019, https://www.gartner.com/en/documents/2487216.

[24] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "Ttpdrill: automatic and accurate extraction of threat actions from unstructured text of cti sources," in *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 103–115, ACM, Orlando, FL, USA, December 2017.

[25] J. Xie, R. Girshick, and A. Farhadi, "Unsupervised deep embedding for clustering analysis," in *Proceedings of the International Conference on Machine Learning*, pp. 478–487, New York City, NY, USA, June 2016.

[26] P. Owezarski, "Unsupervised classification and characterization of honeypot attacks," in *Proceedings of the 10th International Conference on Network and Service Management (CNSM) and Workshop*, pp. 10–18, IEEE, Rio de Janeiro, Brazil, November 2014.

[27] H. Cho, S. Lee, B. Kim, Y. Shin, and T. Lee, "The study of prediction of same attack group by comparing similarity of domain," in *Proceedings of the 2015 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1220–1222, IEEE, Jeju Island, South Korea, October 2015.

[28] V. Ghiëtte, H. Griffioen, and C. Doerr, "Fingerprinting tooling used for {SSH} compromisation attempts," in *Proceedings of the 22nd International Symposium on Research in Attacks, Intrusions and Defenses {RAID} 2019*, pp. 61–71, Beijing, China, September 2019.

[29] MaxMind, Geolite, 2020, https://dev.maxmind.com/geoip/geoip2/geolite2/.

[30] I. Neamtiu, J. S. Foster, and M. Hicks, "Understanding source code evolution using abstract syntax tree matching," in *Proceedings of the 2005 International Workshop on Mining Software Repositories*, pp. 1–5, Saint Louis, MO, USA, May 2005.

[31] Ipipdotnet, Ipdb-Python, 2019, https://github.com/ipipdotnet/ipdb-python.

[32] Chronicle Security, Virustotal, 2019, https://www.virustotal.com/gui/home/url.

[33] RiskAnalytics, Malware Domain Blocklist, 2019, http://www.malwaredomains.com/.

[34] Dr.Web, Web Link Scanner, 2019, https://free.drweb.cn/.

[35] RTBAsia, Rtbasia Api, 2019, https://www.rtbasia.com/ip-lab.

[36] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967.

[37] N. Dunford and J. T. Schwartz, *Linear Operators Part I: General Theory*, Vol. 243, Interscience Publishers, New York, NY, USA, 1958.

[38] K. S. Kim, C. Park, and J. Lee, "Internet home network electrical appliance control on the internet with the upnp expansion," in *Proceedings of the 2006 International Conference on Hybrid Information Technology*, pp. 629–634, IEEE, Jeju Island, South Korea, November 2006.

[39] ATT&CK Matrix, Tactic, 2020, https://attack.mitre.org/.

[40] E. Rendón, I. Abundez, A. Arizmendi, and E. M. Quiroz, "Internal versus external cluster validation indexes," *International Journal of Computers and Communications*, vol. 5, pp. 27–34, 2011.

[41] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics—Theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[42] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53–65, 1987.

[43] D. L. Davies and D. W. Bouldin, "A cluster separation measure," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[44] Y. Cheng, "Mean shift, mode seeking, and clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, pp. 790–799, 1995.

[45] J. A. Hartigan and M. A. Wong, "Algorithm as 136: a k-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[46] MITRE, Apt Groups, 2020, https://attack.mitre.org/groups/.