symmetry

MDPI

Article

IoT Application-Layer Protocol Vulnerability Detection using Reverse Engineering

Jian-Zhen Luo, Chun Shan *, Jun Cai and Yan Liu

School of Electronic and Information, Guangdong Polytechnic Normal University, Guangzhou 510665, China; luojz@gpnu.edu.cn (J.-Z.L.); caijun@gpnu.edu.cn (J.C.); liuyan_sysu@163.com (Y.L.)

* Correspondence: shanchun@gpnu.edu.cn; Tel.: +86-136-0902-4077

Received: 25 September 2018; Accepted: 26 October 2018; Published: 1 November 2018

Abstract: Fuzzing is regarded as the most promising method for protocol vulnerabilities discovering in network security of Internet of Things (IoT). However, one fatal drawback of existing fuzzing methods is that a huge number of test files are required to maintain a high test coverage. In this paper, a novel method based on protocol reverse engineering is proposed to reduce the amount of test files for fuzzing. The proposed method uses techniques in the field of protocol reverse engineering to identify message formats of IoT application-layer protocol and create test files by generating messages with error fields according to message formats. The protocol message treated as a sequence of bytes is assumed to obey a statistic process with change-points indicating the boundaries of message fields. Then, a multi-change-point detection procedure is introduced to identify change-points of byte sequences according to their statistic properties and divide them into segments according to their change-points. The message segments are further processed via a position-based occurrence probability test analysis to identify keyword fields, data fields and uncertain fields. Finally, a message generation procedure with mutation operation on message fields is applied to construct test files for fuzzing test. The results show that the proposed method can effectively find out the message fields and significantly reduce the amount of test files for fuzzing test.

Keywords: vulnerability detection; IoT security; change-point detection; protocol reverse engineering

1. Introduction

Fuzzing is a widely used security technique for discovering vulnerability in network protocol by sending a series of test files with random or fault data to software system implementing specific protocol and observing software exceptions to detect vulnerabilities within the protocol.

Currently, there exist mainly two kinds of fuzzing techniques, i.e., mutation-based and generation-based fuzzing [1]. The former generates test files by injecting random or fault data into sample messages (message is the basic data unit exchanged between processes of application-layer protocol), while the latter constructs fault-injected messages as test files based on specific protocol specification. The mutation-based fuzzing emits a fatal problem that too many fault-injected messages are required to maintain a high test coverage, such as FileFuzz (http://www.securiteam.com/tools/5PP051FGUE.html) and SPIKEfile (https://www.ee.oulu.fi/research/ouspg/SPIKEfile). However, the amount of fault-injected files is 256^L, where L is the power of sample message's length, and it would take tremendously long time to handle so great amount of fault-injected test files especially when L is large. Actually, a protocol's software system parses inputs by considering their formats and treats any files which does not obey the rule of its format as invalid input, in which case a software system will throw an error and quit before it reaches the fault segment(s). Therefore, many of fault-injected test files are not necessary for successful fuzzing test. The generation-based fuzzing generates test files by considering the format of input messages, such as

PROTOS (https://www.ee.oulu.fi/roles/ouspg/Protos). One advantage of such fuzzing tools is that it reduces the number of test files greatly and introduces nearly no sacrifice on test coverage [2]. However, one has to figure out the message formats and configure generation-based fuzzer accordingly. Currently, the message formats are mainly collected or analyzed in a manual way, which is a time-consuming and error-prone process. To address these issues, protocol reverse engineering [3] is introduced to obtain protocol specification automatically. The protocol specification including message format a set of rules that describe or model a network protocol. Then a field-based fault-injected message generation procedure conducted by the message format is applied to create fuzzing test files.

Protocol message is treated as a byte sequence which could be divided into a sequence of fields. A keyword field usually holds a command, operator or state code of protocol, while a data field is variable subsequence whose content is always changeable, such as the value of some parameters of communication. Generally, message format is recovered by identifying all fields in byte sequences. However, it is hard to locate the boundary of fields and a great challenge to identify fields in message, since a priori information about them is usually not available. The byte sequence of protocol message is supposed to obey an underlying stochastic process in which different fields have their own distribution of symbols and change-points are the boundaries of fields. Apparently, each change-point implies an end point of one field and a start point of another field. With these assumption, our goal of field boundary detection is essentially the problem of multi-change-point detection. This problem can be addressed using change-point detection [4] widely used in time series analysis. When change-points are localized successfully, messages are divided into field sequences. However, the type of fields are still uncertain. Thus, a further inference procedure, named position-based occurrence probability test analysis, is proposed to determine field type(keyword fields and data fields). Firstly, fields with approximate zero-probability distribution are classified as data fields. Then, the rest ones are further processed in a position-based statistic test. Specifically, a reference position would be selected for every field, and each field are tested by binomial test to make sure whether their positions are equal to the reference position with probability 1 given a significance level α. The fields passing these tests are chosen as keyword fields, while the rest ones are considered as uncertain fields.

2. Related Work

Recently, the security and privacy issues for Internet of Things have attracted a lot of research interests [5–10]. In particular, the analysis of applications and protocols in real-time network traffic monitoring is a fundamental and critical building block in network management and security systems for IoT infrastructures [11–15]. In this part, we review the recent works in application-layer network protocol vulnerability analysis and detection.

Fuzzing helps protocol vulnerabilities detection to gain higher benefit-to-cost ratio with no or less increasing in computing complexity. It aims to reveal bugs in protocols which would be exploited by adversary to launch attack or activate their malicious code. Currently, research on network protocol fuzzing test is a heat topic in network security. AutoFuzz [16] identified the variable parts of sample messages and fuzzes protocol implementation by sending messages with invalid symbols or messages. AspFuzz [17] leveraged the accessible protocol specifications on RFCs (Request for Comments) to generate fault-injected messages for test files. Then, AspFuzz sent both anomalous and reordered messages to discover vulnerabilities. SecFuzz [18] focused on fuzzing security protocol implementation, but it did not consider the specification of target protocol as well. Zhao et al. [19] used regression finite state machine to infer a state transition diagram of protocol so as to reveal potential vulnerabilities in wireless protocols.

In recent years, a range of works about protocol reverse engineering [20] have been published. Early in 2005, Marshall A. Beddoe held the protocol informatics project [21] and applied bio-informatics algorithms to identify the fields in packets based on alignment algorithms. Cui et al. went further than Beddoe and presented Discoverer [22] to recover protocol message format using both sequence aligning and recursively clustering algorithm. However, Discoverer need some a priori information

about the delimiters used by protocol, such as space and comma, which is used to help tokenization, i.e., breaking message into token sequence. Recently, Tao et al. [23] combines hierarchical clustering algorithm, multi-sequence alignment and Bayesian decision model to determine the field boundary of binary protocol in bit granularity. Chen et al. [24] introduce deep learning algorithm to analyze mobile applications. Xiao et al. [25] propose a method based on heuristic rule to reverse analysis of the incomplete flow. In our approach, we make no assumption about the delimiters. We treat the byte sequence of message as a stochastic process and detect field boundaries according to their statistical properties.

As a paralleled method to understand the unknown protocols, binary analysis-based techniques, such as Polyglot [26], Tupni [27], AutoFormat [28], Prospex [29] Dispatcher [30] and so on, also draw much research attention. They are practical in some special scenarios where binary codes are available and executable in a specific sandbox-like environment. Moreover, binary analysis method would fail if programs make use of some confusion techniques like obfuscation to keep themselves away from being reverse-engineered.

As in many other security application domains [31–36], data mining and machine learning techniques have been widely adopted in the domain of IoT security and IoT traffic analysis. One of the key challenges is the data privacy problem, especially in collaborative and cloud-based learning scenarios. Several recent studies have proposed novel data privacy preserving approaches for addressing the problem [37–42].

3. Problem Formulation

Suppose that the alphabet used by protocol messages is defined as $\Sigma = \left\{0\text{x}00, 0\text{x}01, 0\text{x}02, ..., 0\text{x}FF\right\}$. A string ω is defined as a finite set of ordered letters in Σ. That is $\omega = a_1a_2...a_n$ ($a_1, a_2, ..., a_n \in \Sigma$). All strings over alphabet Σ forms a super set Σ^*. As a basic data unit used by IoT protocol, protocol message m is essentially strings made up of a sequence of message fields. Thus, we mark message field as $\varrho \in \Sigma^*$.

In this paper, a protocol message is assumed to be a byte sequence undergoing hidden statistical process, denoted as Θ, whose statistical feature would shift on and on when the byte sequence goes from one message field to another. As Θ passes from one field (ϱ_i) to another (ϱ_j), the statistical characteristic would change significantly. Thus, a change-point would occur just in the boundary of two different message fields. Inspired by this observation, the problem of message field identification can be transformed to be a change-point detection issue in the statistical process undergone by protocol message.

Given a string $\omega_o = \overline{x_1...x_n}$, a q-length prefix of the last letter (i.e., x_n) in ω_o is marked as $\mathfrak{T}(\omega_o, q)$, while the set of such prefixes whose lengths are no longer than Q in ω_o is marked as $\mathcal{T}(x_n, Q)$. For instance, $\mathfrak{T}(\overline{x_1...x_4}, 2) = \overline{x_2x_3}$, $\mathfrak{T}(\overline{x_1...x_4}, 3) = \overline{x_1x_2x_3}$, and $\mathcal{T}(\omega_o, Q) = \{\mathfrak{T}(\omega_o, q) : 1 \leq q \leq \min(Q, n-1), Q \in \mathbb{R}\}$.

The prefix conditional probability of $\overline{x_1...x_n}$ is defined as

$$p_n = P\Big(x_n|\mathcal{T}(\overline{x_1...x_n}, n-1)\Big). \tag{1}$$

Let $m = \overline{x_1x_2x_3...}$ to be a Q-order Markov process. Then, the likelihood of x_n given $\overline{x_1, ..., x_{n-1}}$ is

$$P(x_n|\overline{x_1, ..., x_{n-1}}) = P(x_n|\overline{x_{n-Q}x_{n-Q+1}...x_{n-1}}), \tag{2}$$

where $Q \in \mathbb{R}$ and $n > Q$.

Suppose that the byte sequence of protocol message obeys Q-order Markov process, then Equation (1) would be rewritten as follows.

$$p_n = P\Big(x_n|\mathcal{T}(\overline{x_1...x_n}, Q)\Big) = \sum_{q=1}^{\min\{Q,n-1\}} \omega_q P(x_n|\mathfrak{T}(\overline{x_1...x_n}, q)), \quad (3)$$

where ω_q is the weight of $P(x_n|\mathfrak{T}(\overline{x_1...x_n}, q))$. Essentially, ω_q can be regarded as the importance of $\mathfrak{T}(\overline{x_1...x_n}, q)$ for predicting the context of x_n.

The larger q is, the more important it is for $\mathfrak{T}(\overline{x_1...x_n}, q)$ in predicting the context of x_n. For instance, it is much more important for $P(\text{“}e\text{”}|\text{“}xampl\text{”})$ than $P(\text{“}e\text{”}|\text{“}pl\text{”})$ to foresee that the context of “e” is “example” instead of “multiple”. As a result, the weight of ω_q in this paper is defined as

$$\omega_q = \frac{q^2}{\sum_{q'=1}^{\min\{Q,n-1\}} (q')^2}, \quad n > 1. \quad (4)$$

Additionally, $P(x_n|\mathfrak{T}(\overline{x_1...x_n}, q))$ is calculated by

$$P(x_n|\mathfrak{T}(\overline{x_1...x_n}, q)) = \frac{\nu(\overline{x_{n-q}...x_{n-1}x_n})}{\nu(\overline{x_{n-q}...x_{n-1}})}, \quad (5)$$

where $\nu(\omega)$ is the frequency of ω in training dataset $\mathcal{D}$.

As shown in Figure 1, the prefix conditional probability of $\overline{x_1...x_n}$ would be very high when x_n and $\mathfrak{T}(x_n, q)$ locate in the same field, otherwise it would be low.

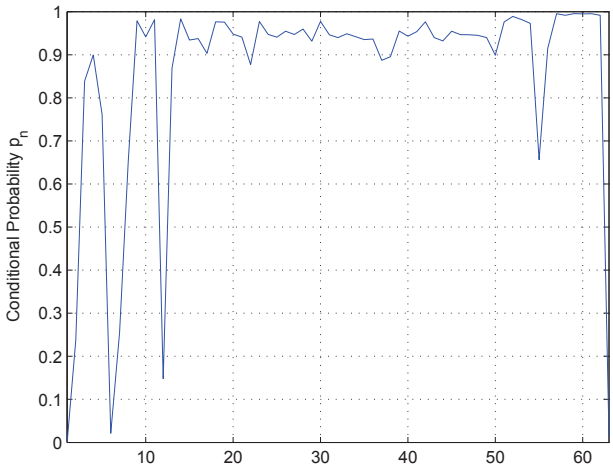

Figure 1. The conditional probability of x_n given $x_1...x_{n-1}$.

3.1. Minmax Formulation for Field Detection

There exist mainly two formulations of change-point detecting problem: Bayesian formulation and minmax formulation. The Bayesian formulation [43] assumes that the change-point γ obeys a prior distribution which is known in prior, while the minmax formulation [44] supposes that the change-point as well as its statistical distribution are unknown to us.

In this paper, the statistical distribution of change-points in protocol message is unknown. As a result, the change-point detection problem should be represented in minmax formulation. Page [45] proposed a cumulative sum (CUSUM) algorithm to implement an optimal solution to minmax formulated problems. Accordingly, a CUSUM-LIKE algorithm is proposed to search fro multiple change-points in this paper. Since the statistic feature of message fields is unknown in prior, the likelihood ratio from post-change probability to pre-change probability, denoted as $L(X_n)$ cannot

be calculated directly by $L(X_n) = f_{\varrho_1}(X_n)/f_{\varrho_0}(X_n)$. Thus, $L(X_n)$ is replaced with a new metric in this paper as

$$C_n = -log(\frac{p_n}{p_{n-1}}), n > 1. \tag{6}$$

Suppose γ is a change-point in a message and x_n is the n-th letter in the message. We assume that the post-change distribution of x_n is $f_{\varrho_1}(X_n)$, while the pre-change distribution of x_n is $f_{\varrho_0}(X_n)$, then prefix conditional probability of x_n, i.e., p_n, would be much less than p_{n-1}, which results in a high and positive value of C_n. When $n < \gamma$, if x_n and x_{n-1} locate in the same field, that is they obey the same distribution, so that $|1 - p_n/p_{n-1}| \leq \epsilon$, where ϵ is a small and positive value, given as a threshold. if x_n and x_{n-1} locate in different fields, that means $n-1$ is also a change-point which should be detected before γ. On the other hand, the value of C_n is likely to be bigger than the given threshold ϵ when $n > \gamma$. As a result, a detection indicator metric which could be regarded recursively for multi-change-point detection should be defined as:

$$V_n = \max\{0, V_{n-1} + C_n\}, n \geq 1, V_0 = 0. \tag{7}$$

The stopping condition can be set as

$$\tau^* = \min\{n \geq 1 : V_n \geq v\}, \tag{8}$$

where v is a threshold of detection indicator.

3.2. Multi-Change-Point Detection

Since the problem of message field identification in this paper is actually a multi-change-point detection problem, the detection procedure has to be extended to a multi-round procedure presented in Section 3.1 and called MultiCUSUM.

A variable χ_n indicating the underlying state of x_n is defined as

$$\chi_n = \begin{cases} 1, & n \in \{n' \geq 1 : V_{n'} \geq v\} \\ 0, & \texttt{otherwise}. \end{cases} \tag{9}$$

Accordingly, the detection statistic is

$$V_n = \begin{cases} \max\{0, u_0 + C_n\}, & \chi_{n-1} = 1, \\ \max\{0, V_{n-1} + C_n\}, & \chi_{n-1} = 0, \end{cases} \tag{10}$$

where u_0 is the initial condition in a new round of detection procedure started once the previous change-point has been found.

The stopping time in the k-th iteration, denoted as τ_k^*, is defined as

$$\tau_k^* = \min\{n > \tau_{k-1}^* : V_n \geq v_k\}, k \geq 1, \tau_0^* = 0, \tag{11}$$

where

$$v_k = \begin{cases} 1 - \zeta, & \tau_{k-1}^* < n < \tau_{k-1}^* + 3 \\ \min\{\rho * \mu_k, 1 - \zeta\}, & n \geq \tau_{k-1}^* + 3 \end{cases} \tag{12}$$

with μ_k as the mean of $\{C_{\tau_{k-1}^*+1}, ..., C_{n-1}\}$ and ρ as the coefficient of μ_k.

3.3. Message Segmenting Algorithm

A message segmenting algorithm, as shown in Algorithm 1, is proposed to segment protocol message m into a set of message fields. In Algorithm 1, the message m consists of a set of All messages associated with a specific protocol in $\mathcal{D}$ are concatenated one by one to form a new message m according to their appearance time. Then, a Q-depth suffix trie $\mathbf{T}$ is built to store sub-strings of m with max length of $Q+1$ (line 1). The prefix conditional probability p_n is calculated according to Equation (3) (line 2) to enable the multi-change-point detection procedures (MultiCUSUM()). The identified change-points are put into $\mathcal{P}_1$ (line 3).

Algorithm 1 Message Segmenting Algorithm

Input: Message $m = \overline{x_1 ... x_N}$
Output: Segment set Ω

1: $\mathbf{T} \leftarrow$ QSufTrie(m); # Creating Q-depth suffix trie
2: $\mathbf{P} \leftarrow$ condProb(m,$\mathbf{T}$); # Compute the conditional probabilities: $\mathbf{P} = \{p_n : n = 1, ..., N\}$
3: $\mathcal{P}_1 \leftarrow$ MultiCUSUM(m,$\mathbf{P}$); # Change-point detection, $\mathcal{P}_1$ is the change-point set
4: $m^R \leftarrow \overline{x_N x_{N-1} ... x_1}$; # Reverse the message stream
5: $\mathbf{T}^R \leftarrow$ QSufTrie(m^R);
6: $\mathbf{P}^R \leftarrow$ condProb(m^R,$\mathbf{T}^R$);
7: $\mathcal{P}_2 \leftarrow$ MultiCUSUM(m^R,$\mathbf{P}^R$);
8: $\mathcal{P} \leftarrow \mathcal{P}_1 \cup \mathcal{P}_2$
9: $\Omega \leftarrow$ MsgSeg(m,$\mathcal{P}$);

Actually, not all change-points are not so sensitive to the prefix conditional probability of $P(x_n|\overline{x_1 ... x_{n-1}})$ to be detected by the aforementioned procedure, instead they are more sensitive to the postfix conditional probability of $P(x_n|\overline{x_{n+1} ... x_N})$ which is essentially the prefix conditional probability $x - n$ in a special string that is the reverse-order of original message. Therefore, we reverse the letter order of m (i.e., $m^R = x_N x_{N-1} ... x_1$) and perform the same detection procedure again on m^R to search for such type of change-points (line 4∼7) and put the results in $\mathcal{P}_2$.

Finally, the two sets of change-points are merged by $\mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2$ and the message m is segmented into segments based on the change-points in $\mathcal{P}$(line 8∼9).

4. Inferring Message Fields

4.1. Occurrence Probability Analysis

To relief the burden of position-based statistic test analysis, a pre-processing called occurrence probability analysis is applied to filter out the obvious part of data fields whose occurrence probability is very low. Given a dataset $\mathcal{D}$ and its size of M, and the occurrence probability of a string $\omega \in \Omega$ in $\mathcal{D}$, denoted as $p_D(\omega)$, is defined as the ratio between the amount of messages containing ω, denoted as $\nu_m(\omega)$, and the size of dataset.

The data field is variational and their occurrence probabilities of each value in a data field are always very small, which nearly approaches zero. Therefore, the data field can be found by searching for those string segments whose occurrence probabilities are statistically zero. In this paper, the occurrence probabilities of message segments is assumed to obey binomial distribution and the binomial test in the statistics field is considered to test whether the occurrence probability of each message segment is zero.

Let the hypothesis be

$$H_0 : p_D(\omega) \to 0, \omega \in \Omega, \tag{13}$$

where α is a significance level.

The strings in $\mathcal{F}$ could be chosen as data fields according to

$$\mathcal{F}_d = \left\{ \omega : \frac{\nu_m(\omega)}{M} \leq \alpha, \omega \in \Omega \right\}. \tag{14}$$

4.2. Position-Based Statistic Test Analysis

Apparently, a keyword field would frequently appear in many messages with similar function and its positions are also relatively stable. That means both frequency and position are important features for us to infer keyword fields from segment set $\mathcal{F}$. As a result, a position-based statistic test is introduced to select keyword fields from $\{\omega : \omega \in (\mathcal{F} - \mathcal{F}_d)\}$ by testing the position of segment is fixed or quasi-fixed in messages.

Specifically, four kinds of positions of ω are considered in our scheme. That is

- $P^{\omega,1}$: the distance between the message head and the position of ω in the message.
- $P^{\omega,2}$: the distance between the message tail and the position of ω in the message.
- $P^{\omega,3}$: the distance between the head of a line which containing ω and the position of ω in that line.
- $P^{\omega,4}$: the distance between the tail of a line which containing ω and the position of ω in that line.

Let $P^{\omega,r} = (p_1^{\omega,r}, ..., p_n^{\omega,r}), r \in \{1,2,3,4\}$ and define the support rate of $p_i^{\omega,r}$, marked as $N(p_i^{\omega,r})$, as the number of $p_i^{\omega,r}$ in $\mathcal{D}$. Based on binomial test (see Section 4.1), the keyword fields are chosen by

$$\mathcal{F}_k = \left\{ \omega : \frac{\max\limits_{i,r} \left\{ N(p_i^{\omega,r}) \right\}}{\sum_{r,i} N(p_i^{\omega,r})} > 1 - \alpha, \omega \in \mathcal{F} - \mathcal{F}_d \right\}, \tag{15}$$

given α as the significance level.

Equation (15) infers keywords whose positions are fixed by searching for segments satisfying $\max_{i,r}\{N(p_i^{\omega,r})\}$. It has good performance on those ω which have one dominated position. For instance, "GET" in HTTP messages has one dominated position, i.e., in the head of a request message. However, some other keywords have more than one dominated position, and there are multiple peaks in $N(p_i^{\omega,r})$.

Aiming to address multi-peak issue, an algorithm (called MDL-PTA) based on the minimal description length (MDL) [46] criteria is introduced to enable the position-based statistic test analysis, as shown in Algorithm 2.

k reference positions, $B_k = \{b_1, ..., b_k\}$, whose support rates are the first k top values in $\{N(p_j^{\omega,r}) : p_j^{\omega,r} \in P^{\omega,r}\}$ (line 5) are selected for each ω, and $P^{\omega,r}$ is divided into k clusters, $C_k = \{c(b_1), ..., c(b_k)\}$, according to the distance between $p_j^{\omega,r}$ and reference position b_m, $m = 1, 2, ..., k$ (line 6).

The entropy of C_k is calculated through following equation:

$$E_k = -\sum_{i=1}^{k} \left(\frac{|c(b_i)|}{\sum_{j=1}^{k} |c(b_j)|} \log \frac{|c(b_i)|}{\sum_{j=1}^{k} |c(b_j)|} \right). \tag{16}$$

The model complexity of C_k is $(\log k)/2$ and the sum of description length of C_k is calculated in line 7, that is

$$L_k = E_k + \frac{\log k}{2}. \tag{17}$$

The k-th model in the model set Ψ is represented as $\{B_k, C_k, L_k\}$. The optimal model with minimal description length would be selected from Ψ (line 11). Apparently, the computation complexity would be very high if all models in Ψ are considered. Meanwhile, a keyword should not have lots of reference positions. As a result, only the top K models in Ψ are considered in Algorithm 2 (line 4∼10).

Algorithm 2 MDL-PTA Algorithm

Input: K, $\mathcal{D}$ and $\omega \in (\mathcal{F} - \mathcal{F}_d)$
Output: **true** if ω is a keyword field, or **false** otherwise.

$\Psi \leftarrow \{\}$
$P^{\omega,r} \leftarrow$ GetPos($\mathcal{D}$,ω) # Get positions of ω
$\mathcal{N} \leftarrow \{N(p_j^{\omega,r}) : p_j^{\omega,r} \in P^{\omega,r}\}$
for i=1 **to** K **do**
  $B_i \leftarrow$TopK($\mathcal{N}$,i) # Get reference positions
  $C_i \leftarrow$Cluster($P^{\omega,r}$,B_i) # Cluster $P^{\omega,r}$
  $L_i \leftarrow$CalDLen(C_i) # Compute the description length of C_i
  $\psi_i \leftarrow \{B_i, C_i, L_i\}$
  $\Psi \leftarrow \Psi \cup \{\psi_i\}$
end for
$\psi^* \leftarrow$ minDLen(Ψ) # Get the model with minimal description length
for all $b_i \in B^*$ **do**
  res$\leftarrow$TestPos(b_i,$c^*(b_i)$) # Check whether b_i satisfies Eq. (19)
  if res==**true then**
    return true
  end if
end for
return false

The optimal model chosen by the Algorithm 2 is $\psi^* = \{C^*, B^*, L^*\}$. For each reference position $b_i \in B^*$ in $C^* = \{c^*(b_1), ..., c^*(b_{k^*})\}$, the following hypothesis is tested via binomial test:

$$H_0 : P(x = b_i : x \in c^*(b_i)) \rightarrow 1, \tag{18}$$

where $b_i \in B^*$ and $i = 1, ..., k^*$.

The accept condition is

$$\left\{ b \in B^* : \frac{N(b)}{\sum_{x \in c^*(b)} N(x)} > 1 - \alpha \right\}, \tag{19}$$

The segment set passing hypothesis test is regarded as keyword set $\mathcal{F}_k$, and the rest ones are uncertain fields.

The inferred message fields would be further refined and some semantic information of message fields would be determined. Specifically, continuous segments of data (or uncertain) fields would be merged into a single segment which is data (or uncertain) field. Regular expressions representing some specific semantic information, such as IP address, File names, URLs, Timestamp and so on, are applied to match the message fields so that some semantic of message fields would be inferred.

5. Evaluation

In this section, experiments are performed to evaluate the effectiveness of the proposed method. The experiments comprise of two parts: message segmentation evaluation and fuzzing test. The proposed message segmentation approach is implemented on a system called QCD-PInfer whose system architecture is shown in Figure 2.

There are totally six typical protocols (HTTP, FTP, SMTP, POP, DNS and QQ) which are widely used in the application-layer are selected to test the effectiveness and efficiency of message segmentation. The recall and precision of keyword inference are shown in Tables 1 and 2. Please note that, the ground truth of keywords are those keywords which are occurred in the test set. Both DNS and QQ are not

taken into account for evaluating the quality of keyword set, since the two are binary protocols and there is no concept of keyword defined in binary protocol.

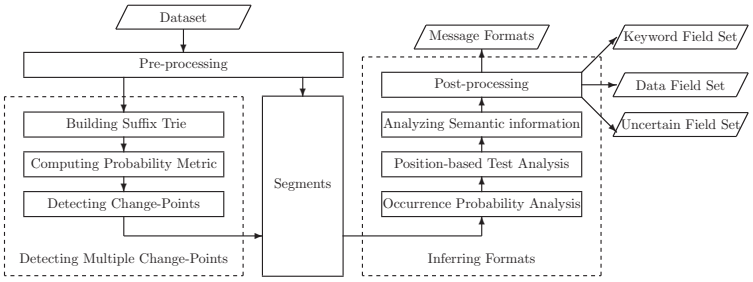

Figure 2. The system architecture of QCD-PInfer.

By comparison, QCD-PInfer has a higher recall rate than Discoverer and PI. In particular, PI's recall rate is much low: the recall rates for HTTP, FTP, SMTP and POP are less than 10%.

Table 1. The recall of keyword inference.

System	HTTP	FTP	SMTP	POP	SSDP	BitTorrent
QCD-PInfer	87.0	92.9	85.7	84.0	74.1	100
Discoverer	78.3	60.7	64.3	40.0	33.3	100
PI	4.4	3.6	7.1	4.0	18.5	50.0

Discoverer is prone to infer too many segments as keywords, so that its precision is much lower than that of the proposed system. Although PI's recall rate is very low, its precision for HTTP and FTP is extremely high. However, PI's precision for other protocols are still very low. It is worth mentioning that PI infers too few keywords, always less than 5 for all protocols being considered.

Table 2. The precision of keyword inference.

System	HTTP	FTP	SMTP	POP	SSDP	BitTorrent
QCD-PInfer	66.7	97.0	35.0	95.8	66.0	66.7
Discoverer	7.2	23.3	19.2	22.8	33.9	5.3
PI	100	100	20.0	16.7	35.6	33.3

The F-scores of the experiment results are shown in Figure 3. The proposed system has the highest F-score for all the six protocols, which means our method performs well in keyword inference.

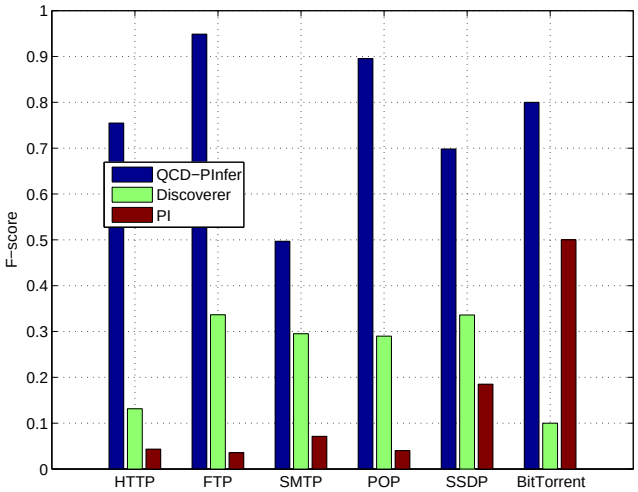

Figure 3. The F-score value of keywords.

In fuzzing test, QCD-PInfer is extended with fuzzing function to implement an automatic fuzzing tool (APREFuzz). APREFuzz can identify vulnerability in a system being tested which is designed to introduce information-centric network into IoT devices to enable their caching capability. The protocol used by target system under testing comprises of 5 type of messages responsible for sending interesting, distributing data, pushing data, responding with target data and responding with no answer, respectively.

Firstly, message fields are identified using QCD-PInfer system, and message format are reconstructed.

Secondly, test files are generated by inserting fault data into one field according to the message format. Please note that, for a real fuzz test, fault data may inserted into more than one field. However, as a proof-of-concept system, APREFuzz considers the scenarios with only one field being fault-injected currently. Actually, it is not difficult to extend the system to consider fault-injected in multiple fields. When inserting the fault data, keyword fields are only replacing with inferred keywords according to message formats, data fields would be replaced by random data, while uncertain fields would be replaced with either inferred keywords or random data. In our experiments, the uncertain fields are treated the same as data fields.

Finally, the target system are treated as a black box and supposed to be unknown to us. APREFuzz sends test files to target system file-by-file and monitors the reactive of target system via analyzing the response.

In our experiments, APREFuzz extracted 7 keyword fields and infers 7 data fields in the sample message. One data field is found that it contains only figures. The amount of inferred keywords is 12. We take 11 abnormal strings into account for inserting fault data into the data fields except the one containing only figures. For the special field that containing only figures, 21 boundary figures are used to be injected. As a result, the amount of fault-injected files generated by APREFuzz is 248 ($=(12+11)\times 7+21\times 1+11\times 6$). On the other hand, FileFuzz generates $393,216$ ($=1.5\times 1024\times 2^8$) fault-injected files by replacing each byte with values from 0x00 to 0xFF. When test files are sent to target system, APREFuzz monitored one exception that the system fails to respond, while FileFuzz monitor none. The exceptions maybe indicates a vulnerability which would be leveraged to launch a DoS attack, or some attacks that would ruin the system's availability. Actually, other tools are needed to analyze the exception deeply and figure out its type and impact. However, that work has surpassed the discussion scope concerned in this paper so that it will not be presented here.

6. Conclusions

The proposed method applies protocol reverse engineering approach to improve IoT protocol fuzzing performance by creating valid and effective test files based on protocol message format and reducing greatly the size of test files. It considers the statistical attributes of message fields to locate their boundaries by searching for change-points in the messages and reconstruct the message format. A CUSUM-LIKE algorithm is presented to address the problem of multi-change-point detection. Additional procedures including occurrence probability test and position test are further employed to classify the message segments into keyword fields , data fields and uncertain fields. The results show that the extracted message formats are useful for generating test files for network protocol fuzzing.

In the future, the proposed APREFuzz with enough improvement based on current version would be a practical and powerful tool to generate test files automatically for fuzzing test carried on IoT protocols or devices to reveal their hidden vulnerabilities. It also would contribute to strengthening the IoT security in effective and efficient way, and even to be a security tool for improving protocol fuzzing in many other types of network.

Author Contributions: Conceptualization, J.-Z.L. and J.C.; methodology, J.-Z.L.; software, J.-Z.L. and C.S.; validation, J.-Z.L., J.C., Y.L. and C.S.; formal analysis, C.S.; investigation, C.S.; resources, J.C.; data curation, Y.L.; writing—original draft preparation, J.-Z.L.; writing—review and editing, J.C. and C.S.; visualization, Y.L.; supervision, J.C.; project administration, J.C.; funding acquisition, J.C.

Funding: This research was funded by National Natural Science Foundation of China (Grant No.: 61702120, 61571141); Natural Science Foundation of Guangdong Province (Grant No.: 2017A030310591, 2014A030313637, 2015A030313672); Department of Education of Guangdong Province (Grant No.: YQ2015105, 2016GCZX006, 2016KQNCX091); Guangdong Provincial Application-oriented Technical Research and Development Special fund project (Grant No.: 2015B010131017); Guangdong Science and Technology Department (Grant No.: 2016A010120010, 2014A010103032, 2017A090905023); Science and Technology Program of Guangzhou (Grant No.: 201604016108).

Acknowledgments: The authors would also like to thank the anonymous reviewers for their valuable comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Munea, T.L.; Lim, H.; Shon, T. Network protocol fuzz testing for information systems and applications: A survey and taxonomy. *Multimed. Tools Appl.* **2016**, *75*, 14745–14757. [CrossRef]
2. Kim, H.C.; Choi, Y.H.; Lee, D.H. Efficient file fuzz testing using automated analysis of binary file format. *J. Syst. Archit.* **2011**, *57*, 259–268. [CrossRef]
3. Duchêne, J.; Le Guernic, C.; Alata, E.; Nicomette, V.; Kaâniche, M. State of the art of network protocol reverse engineering tools. *J. Comput. Virol. Hacking Tech.* **2018**, *14*, 53–68. [CrossRef]
4. Aminikhanghahi, S.; Cook, D.J. A survey of methods for time series change point detection. *Knowl. Inf. Syst.* **2017**, *51*, 339–367. [CrossRef] [PubMed]
5. Yan, H.; Li, X.; Wang, Y.; Jia, C. Centralized Duplicate Removal Video Storage System with Privacy Preservation in IoT. *Sensors* **2018**, *18*, 1814. [CrossRef] [PubMed]
6. Yang, Y.; Zheng, X.; Tang, C. Lightweight distributed secure data management system for health internet of things. *J. Netw. Comput. Appl.* **2017**, *89*, 26–37. [CrossRef]
7. Tan, Q.; Gao, Y.; Shi, J.; Wang, X.; Fang, B.; Tian, Z.H. Towards a Comprehensive Insight into the Eclipse Attacks of Tor Hidden Services. *IEEE Internet Things J.* **2018**. [CrossRef]
8. Wang, Z. A privacy-preserving and accountable authentication protocol for IoT end-devices with weaker identity. *Future Gener. Comput. Syst.* **2018**, *82*, 342–348. [CrossRef]
9. Luo, E.; Bhuiyan, M.Z.A.; Wang, G.; Rahman, M.A.; Wu, J.; Atiquzzaman, M. PrivacyProtector: Privacy-Protected Patient Data Collection in IoT-Based Healthcare Systems. *IEEE Commun. Mag.* **2018**, *56*, 163–168. [CrossRef]
10. Mao, Y.; Li, J.; Chen, M.R.; Liu, J.; Xie, C.; Zhan, Y. Fully secure fuzzy identity-based encryption for secure IoT communications. *Comput. Stand. Interfaces* **2016**, *44*, 117–121. [CrossRef]
11. Liu, Q.; Wang, G.; Liu, X.; Peng, T.; Wu, J. Achieving reliable and secure services in cloud computing environments. *Comput. Electr. Eng.* **2017**, *59*, 153–164. [CrossRef]
12. Chen, Z.; Peng, L.; Gao, C.; Yang, B.; Chen, Y.; Li, J. Flexible neural trees based early stage identification for IP traffic. *Soft Comput.* **2017**, *21*, 2035–2046. [CrossRef]
13. Meng, W.; Tischhauser, E.W.; Wang, Q.; Wang, Y.; Han, J. When Intrusion Detection Meets Blockchain Technology: A Review. *IEEE Access* **2018**, *6*, 10179–10188. [CrossRef]
14. Zhou, Z.; Dong, M.; Ota, K.; Wang, G.; Yang, L.T. Energy-Efficient Resource Allocation for D2D Communications Underlaying Cloud-RAN-Based LTE-A Networks. *IEEE Internet Things J.* **2016**, *3*, 428–438. [CrossRef]
15. Cai, J.; Wang, Y.; Liu, Y.; Luo, J.Z.; Wei, W.; Xu, X. Enhancing network capacity by weakening community structure in scale-free network. *Future Gener. Comput. Syst.* **2018**, *87*, 765–771. [CrossRef]
16. Gorbunov, S.; Rosenbloom, A. AutoFuzz: Automated Network Protocol Fuzzing Framework. *Int. J. Comput. Sci. Netw. Secur.* **2010**, *10*, 239–245.
17. Kitagawa, T.; Hanaoka, M.; Kono, K. AspFuzz: A state-aware protocol fuzzer based on application-layer protocols. In Proceedings of the IEEE Symposium on Computers and Communications, Riccione, Italy, 22–25 June 2010; pp. 202–208.
18. Tsankov, P.; Dashti, M.T.; Basin, D. SecFuzz: Fuzz-testing security protocols. In Proceedings of the International Workshop on Automation of Software Test, Zurich, Switzerland, 2–3 June 2012; pp. 1–7.
19. Zhao, J.; Chen, S.; Liang, S.; Cui, B.; Song, X. RFSM-Fuzzing a Smart Fuzzing Algorithm Based on Regression FSM. In Proceedings of the Eighth International Conference on P2p, Parallel, Grid, Cloud and Internet Computing, Compiegne, France, 28–30 October 2013; pp. 380–386.

20. Narayan, J.; Shukla, S.K.; Clancy, T.C. A survey of automatic protocol reverse engineering tools. *ACM Comput. Surv. (CSUR)* **2016**, *48*, 40. [CrossRef]
21. Beddoe, M.A. Network Protocol Analysis Using Bioinformatics Algorithms. 2004. Available online: http://www.4tphi.net/~awalters/PI/pi.pdf (accessed on 28 October 2018).
22. Cui, W.; Kannan, J.; Wang, H.J. Discoverer: Automatic protocol reverse engineering from network traces. In Proceedings of the 16th USENIX Security Symposium on USENIX Security Symposium, Boston, MA, USA, 6–10 August 2007; USENIX Association: Berkeley, CA, USA, 2007; pp. 1–14.
23. Tao, S.; Yu, H.; Li, Q. Bit-oriented format extraction approach for automatic binary protocol reverse engineering. *IET Commun.* **2016**, *10*, 709–716. [CrossRef]
24. Zhengyang, C.; Bowen, Y.; Yu, Z.; Jianzhong, Z.; Jingdong, X. Automatic Mobile Application Traffic Identification by Convolutional Neural Networks. In Proceedings of the IEEE Trustcom/BigDataSE/SPA, Tianjin, China, 23–26 August 2016; pp. 301–307.
25. Xiao, M.M.; Zhang, S.L.; Luo, Y.P. Automatic network protocol message format analysis. *J. Intell. Fuzzy Syst.* **2016**, *31*, 2271–2279. [CrossRef]
26. Caballero, J.; Yin, H.; Liang, Z.; Song, D. Polyglot: Automatic extraction of protocol message format using dynamic binary analysis. In Proceedings of the 14th ACM conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 Novemver 2007; ACM: New York, NY, USA, 2007; pp. 317–329.
27. Cui, W.; Peinado, M.; Chen, K.; Wang, H.J.; Irun-Briz, L. Tupni: Automatic reverse engineering of input formats. In Proceedings of the 15th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 27–31 October 2008; ACM: New York, NY, USA, 2008; pp. 391–402.
28. Lin, Z.; Jiang, X.; Xu, D.; Zhang, X. Automatic Protocol Format Reverse Engineering through Context-Aware Monitored Execution. *NDSS* **2008**, *8*, 1–15.
29. Comparetti, P.; Wondracek, G.; Kruegel, C.; Kirda, E. Prospex: Protocol Specification Extraction. In Proceedings of the 2009 30th IEEE Symposium on Security and Privacy, Berkeley, CA, USA, 17–30 May 2009; pp. 110–125.
30. Caballero, J.; Poosankam, P.; Kreibich, C.; Song, D. Dispatcher: Enabling active botnet infiltration using automatic protocol reverse-engineering. In Proceedings of the 16th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 9–13 November 2009; ACM: New York, NY, USA, 2009; pp. 621–634.
31. Meng, W.; Wang, Y.; Wong, D.S.; Wen, S.; Xiang, Y. TouchWB: Touch behavioral user authentication based on web browsing on smartphones. *J. Netw. Comput. Appl.* **2018**, *117*, 1–9. [CrossRef]
32. Li, J.; Sun, L.; Yan, Q.; Li, Z.; Srisa-an, W.; Ye, H. Significant Permission Identification for Machine Learning Based Android Malware Detection. *IEEE Trans. Ind. Inform.* **2018**. [CrossRef]
33. Liu, Y.; Ling, J.; Liu, Z.; Shen, J.; Gao, C. Finger Vein Secure Biometric Template Generation Based on Deep Learning. *Soft Comput.* **2018**, *22*, 2257–2265. [CrossRef]
34. Yuan, C.; Li, X.; Wu, Q.; Li, J.; Sun, X. Fingerprint Liveness Detection from Different Fingerprint Materials Using Convolutional Neural Network and Principal Component Analysis. *CMC-Comput. Mater. Contin.* **2017**, *53*, 357–371.
35. Meng, W.; Jiang, L.; Wang, Y.; Li, J.; Zhang, J.; Xiang, Y. JFCGuard: Detecting juice filming charging attack via processor usage analysis on smartphones. *Comput. Secur.* **2018**, *76*, 252–264. [CrossRef]
36. Chen, S.; Wang, G.; Yan, G.; Xie, D. Multi-imensional fuzzy trust evaluation for mobile social networks based on dynamic community structures. *Concurr. Comput. Pract. Exp.* **2017**, *29*, e3901. [CrossRef]
37. Li, P.; Li, J.; Huang, Z.; Gao, C.Z.; Chen, W.B.; Chen, K. Privacy-preserving outsourced classification in cloud computing. *Clust. Comput.* **2017**. [CrossRef]
38. Li, P.; Li, J.; Huang, Z.; Li, T.; Gao, C.Z.; Yiu, S.M.; Chen, K. Multi-key privacy-preserving deep learning in cloud computing. *Future Gener. Comput. Syst.* **2017**, *74*, 76–85. [CrossRef]
39. Li, J.; Zhang, Y.; Chen, X.; Xiang, Y. Secure attribute-based data sharing for resource-limited users in cloud computing. *Comput. Secur.* **2018**, *72*, 1–12. [CrossRef]
40. Gao, C.Z.; Cheng, Q.; Li, X.; Xia, S.B. Cloud-assisted privacy-preserving profile-matching scheme under multiple keys in mobile social network. *Clust. Comput.* **2018**. [CrossRef]
41. Luo, E.; Liu, Q.; Abawajy, J.H.; Wang, G. Privacy-preserving multi-hop profile-matching protocol for proximity mobile social networks. *Future Gener. Comput. Syst.* **2017**, *68*, 222–233. [CrossRef]

42. Zhi Gao, C.; Cheng, Q.; He, P.; Susilo, W.; Li, J. Privacy-preserving Naive Bayes classifiers secure against the substitution-then-comparison attack. *Inf. Sci.* **2018**, *444*, 72–88.
43. Shiryaev, A. On Optimum Methods in Quickest Detection Problems. *Theory Probab. Appl.* **1963**, *8*, 22–46. [CrossRef]
44. Lorden, G. Procedures for Reacting to a Change in Distribution. *Ann. Math. Stat.* **1971**, *42*, 1897–1908. [CrossRef]
45. Page, E.S. Continuous Inspection Schemes. *Biometrika* **1954**, *41*, 100–115. [CrossRef]
46. Rissanen, J. Universal coding, information, prediction, and estimation. *IEEE Trans. Inf. Theory* **1984**, *30*, 629–636. [CrossRef]