M-ISDS: A Mobilized Intrusion and Spam Detection System

Yuyang Li*

School of Computer Science and Technology, Xidian University, Xi'an, 710126, China

ABSTRACT: As the world strides into the digital world, cybersecurity has become an indispensable part of connected devices. Although we have developed cybersecurity measures that can effectively defend against malicious software, we don't have an accurate solution against attacks like social engineering attack, scam calls or phishing. In this work, a novel detection system called m-isds(mobilized intrusion and spam detection system) architecture is proposed, borrowing ideas from the widely utilized advanced hybrid intrusion detection system(ids), combining with some novel concepts including machine learning, advanced hashing technologies and pattern-matching technologies that are secure and cryptographically safe to provide a solution to the proposed system with low false-positive rate and privacy infringement while remaining responsive and flexible against all types of attacks. The system aims to scan the content of the whole terminal on the fly, not only containing and defending against the threat of malicious softwares but also alerting the user of possible scams and spams, bringing the security goal of mobile devices to a whole new level.

1. INTRODUCTION

As the digital age progresses, cybersecurity has become more of a major issue for people of this era. Although manufacturers and developers have made computing and interconnecting more and more secure, Security vulnerabilities are still being breached and exploited, while threats like social engineering attacks and DNS hijacking remain as powerful as always, compromising either user data or leading users towards possible monetary loss. Hence, to protect users from cyberattacks and safeguard their digital and physical assets, we could implement a cross-platform intelligent system that could make valid detections and send out accurate warnings to the end-user while the user's privacy is preserved.

The subjects that this paper will discuss will cover how we can develop our system upon the Traditional Ideas of IDS [1]. Also, pattern-matching and secure hashing functions will be discussed, as well as machine learning. The problems that this paper will go into and analyze contains: how to accurately identify scam information distributed by either phone calls and messages or by the internet, and how to effectively extract the identified information without infringing the privacy of the end-users. The usage of machine learning models and data mining in the field of anti-scam. This paper will go into each section of the system, then the work will discuss the questions by comparing against different hashing functions and comparing the algorithms of each field and people's acceptance of the system.

2. SYSTEM-WIDE DETECTION AND SNIFFING

In this work, as shown in Figure 1, an architecture that composes of Preprocessing module, Decision Support & Processing Unit is proposed. In the following segments, details and existing algorithms will be compared and the appropriate ones will be chosen.

Furthermore, since the system relies on databases of known malicious data, the decision support system would often establish an encrypted connection to the trusted authority for the system to function. In the process, the user's privacy is protected through multiple measures, as discussed in the passages below.

*Corresponding author. Email: 19030100050@stu.xidian.edu.cn

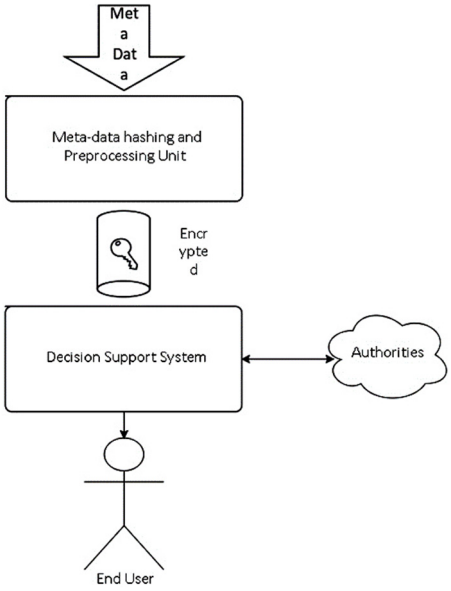

Figure1. System architecture

The system utilizes two kinds of sniffing, namely the Anomaly Detection and Signature Detection, enabling the hybrid system to identify as many threats as possible. The functions of the sub-units of the sniffing and detection will be discussed as follows.

2.1. Unit overview

The Unit consists of two parts, namely The Hashing and Preprocessing subunit and the Anomaly Detection subunit, the data that these two Units output are kept in a Secure Database, as shown in Figure.2 below.

The whole process is computed solely locally to preserve user privacy.

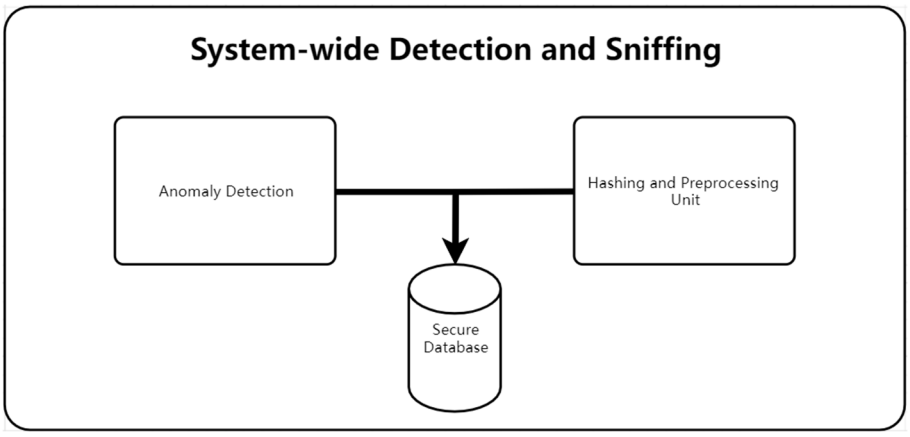

Figure2. System-wide Detection and Sniffing Unit Overview

2.2. Hashing and preprocessing unit

Since the user data Composes of multiple data sources, the system seeks to simplify the whole process by securely hashing and storing the meta data of various kinds, as shown in Figure 3.

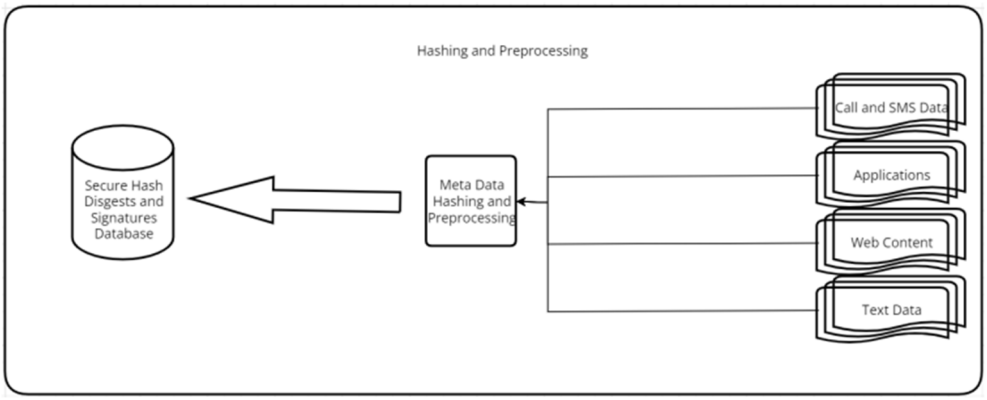

Figure3. Hashing and Preprocessing Unit Architecture

2.2.1. Preprocessing phone call and SMS metadata

To preserve user privacy, the meta data doesn't contain the information of the acoustic data log of the phone call. But rather, it performs a simple gathering of data like the date and time of the call, the duration of the call and the IMEI of the calling device. In terms of SMS data, the mobile number that the particular message has been sent or received will be gathered, as well as the timestamp of the SMS. The data will be encrypted and stored locally, then it will be used in the sections discussed below.

2.2.2. Preprocessing application metadata into signatures

For applications, we can implement the already existing anti-virus measurements to extract their metadata. The typical means for processing them is to use hashing algorithms. Considering that some of the past "secure hashing algorithms" like MD5[2] and SHA1[3] have been cracked (collisions were found) and thus could be used to make malicious software's digital signature disguise as those from a typical, non-threatening software, we must turn to the more secure functions like the SHA-2 families and its successors, or the BLAKE families of faster hashing functions. Among them, we must consider the digest size, which could imply the resistance level of the function since the amount of malicious software is significant, the processing power that the function requires should also be taken into consideration, finally, we should consider which hashing function is used the most for existing malware signature databases.

The existing databases use SHA-256 the most, since the 256-bit space output space achieves a rather higher collision resistance, and the collision of SHA256 has yet to be found. We could perform hashing by SHA-256 algorithms on applications by decompressing the application package in a sandbox and performing SHA-256 inside to extract SHA-256 Digest Data [4].

2.2.3. deconstructing web contents and hashing

Although almost all browsers could send out alerts when the user visits websites with vulnerabilities like invalid digital certificates or outdated protocols, the web contents are not checked, so certain phishing sites can still be a threat to the end user.

We could deconstruct the contents of the web, extract the photos from the source code, then perform a perceptual hashing process that converts the image into a series of values not by the precise values of pixels in the image but based on the number of features of the image.

The system computes these hashes by using an embedding network to produce image descriptors and then converting those descriptors to integers using a Hyperplane LSH (Locality Sensitivity Hashing) process. This process ensures that different images produce different hashes.

The embedding network represents images as real-valued vectors and ensures that perceptually and semantically similar images have close descriptors in the sense of angular distance or cosine similarity. Perceptually and semantically different images have descriptors farther apart, which results in larger angular distances. The Hyperplane LSH process then converts descriptors to unique hash values as integers.

For all images processed by the above system, regardless of resolution and quality, each image must have a unique hash for the content of the image.

The main purpose of the hash is to ensure that identical and visually similar images result in the same hash, and images that are different from one another result in different hashes. For example, an image that has been slightly cropped or resized should be considered identical to its original and have the same hash [5].

2.2.4. Textual data preprocessing

In some spam sites or SMS, the words are jumbled to avoid simple filters, which enables them to avoid censorship and detection, while human beings can still read them effectively. This is called the (The Cmabrigde Uinervtisy Effect). To defend against these attacks, we have to utilize effective word recognition tools to reconstruct the jumbled words to lay the ground stone for valid text classification.

To address The Cmabrigde Uinervtisy Effect, the system utilizes advanced robust word recognition neural network – the scRNN Neural network [6] to reconstruct the misspelled text into the original spelling, through jumbling, Deleting, and inserting.

To tell these texts whether they are malicious, the system first breaks down words into little components while undesirable symbols like the html tags or punctuations are removed. Whitespace tokenization is occupied in the system, outputting continuous words. Then the data gets through a stemming phase, which reduces words to their fundamental meanings. Then the data gets through the process of lemmatization, which employs lexical analysis, eliminating morphological and lexical workarounds that may exist in the data. To complete the whole process, The Language Tool Kit (NLTK) module is utilized.

Furthermore, to prevent malicious software from reading the Cleartext data, the Textual data is transformed into vectors through the Word2Vec model, where each word in the corpus is allocated a matching vector in the space.[7]

2.2.5. Storing hashing digests and signatures Securely

The data the system got from the above must be stored securely, making sure that they could be passed through to the Decision Support System with its integrity intact.

Since the hashing data happens quite a lot when the device is used, the hashing data flow could require a rather large bandwidth. Thus, the system could use symmetrical encryption on the database itself, while the Initial Vector and the secret key that the algorithm need could be encrypted with asymmetric encryption algorithms. This would ensure that the Data can be kept securely yet efficiently.

2.3. Anomaly detection unit

Anomaly detection is an approach to detecting intrusions by first learning the characteristics of normal activity. Then systems are designed to detect anything that deviates from normal activity.

Hence, the anomaly-based detection unit can detect new, unforeseen vulnerabilities and variants of known attacks. It is the other part of the System-wide Detection and Sniffing module that mainly monitors the user profile gathered directly from the mobile device including all logs from various services.

2.3.1. File logs

The Unit would log the system's behavior by monitoring the status of how the files in the system are being accessed (read, write, executed), the logs are then encrypted and stored into the database.

2.3.2. Networking anomaly detection

The networking behavioral data is also logged, the data structure composes of both the Protocol type (ICMP UDP TCP) and Service types (http TELNET), as well as the connection durations and source and destination addresses.

The data is collected and is went through simple filtering, while the whole computing is done at the Decision support system.

The simple filtering process includes sensing attacks like DNS Redirections and ARP spoofing attacks. For example, when the system senses that the network has one or more MAC addresses that appear more than once, the system will send out alerts and end the connection at once.

The more sophisticated attacks would be identified by the Decision Support system discussed below.

2.3.3. Storing logs and data securely

Similar to storing hashing data and signatures, the data in this Unit will be kept securely in an encrypted database.

3. DECISION SUPPORT SYSTEM

The Decision Support system focuses on Processing the various kinds of data gathered in the Preprocessing Unit, the data will be decided whether to be processed locally or to be sent to Cloud-based Regulators and examined on the cloud, where the response will aid the decision making process.

To protect privacy, the private set interchanging (PSI) protocol and Threshold Secret Sharing are used, the details will be discussed below.

The system composes of the following parts: The Known Malicious Data Database which performs its sync when it is connected to the internet, the Pattern Matching and Classification process that determines whether the final processing will be done locally, then the Communication sub-unit that queries the Trusted Regulatory Authorities and get the decisions from the results returned from the cloud.

At last, the decision is made, the alarms are sounded, and action will take place.

The overall architecture is shown below in Figure.4.

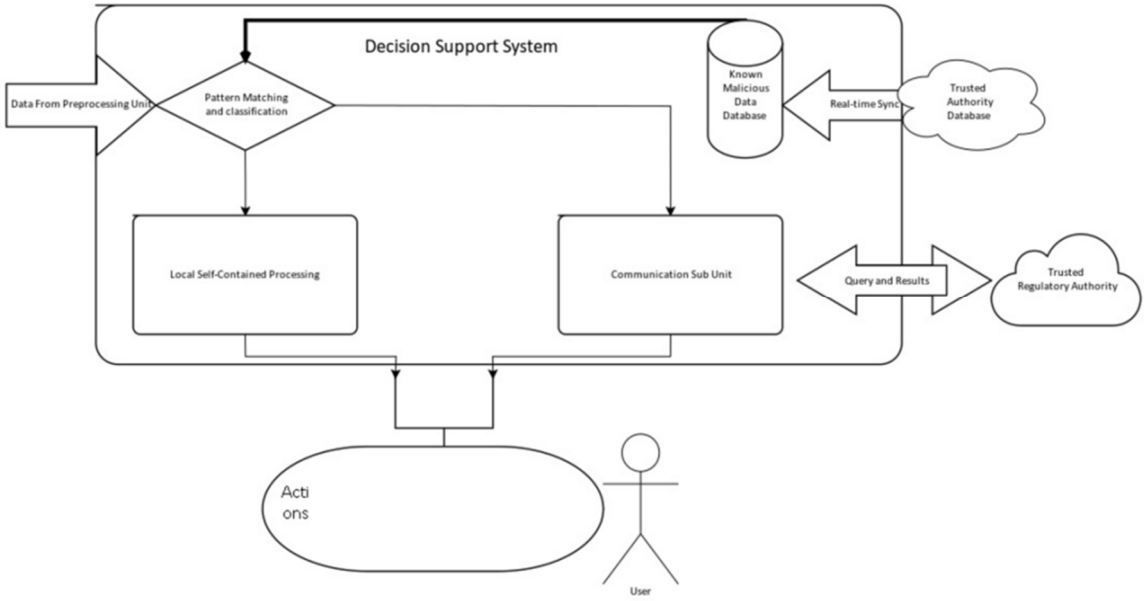

Figure.4 Decision Support System Architecture

3.1. Known malicious data database construction

The Database depends on trusted authorities to provide, since it is them that could gather a large amount of all sorts of data including malicious pictures, spam caller data and SMS, harmful websites data, and also the signatures of malicious applications.

Since the integrity and confidentiality of the database are essential, the system encrypts the database, in this instance, the encryption is backed up by elliptic curve cryptography. Since the mobile device is limited in terms of processing power, it will be a challenging job for the device to apply a robust detection model with a low false-positive rate, some server trained classifiers catered to certain fields will be stored in the database and kept on a frequent sync with the servers.

3.1.1. Call and SMS meta data

Given a dataset composes of the caller's IMEI, Call Duration and Affiliation from the Service Provider, the Service Provider could train a neural network to flag the corresponding callers which have abnormal activities and warn the end user when normal users do pick up the calls. A similar operation could also be performed upon text messages. By using the dataset from [8]. We could train a neural network that utilizes the Resnet [9] to achieve a 90% accuracy while achieving a low false positive rate of 2.4%.

This method ensures the service providers could flag the spiteful callers effectively.

Furthermore, since the content of scam text messages always contains certain keywords, the provider could extract the word vectors from spam texts and store them through hashing functions and store them on the cloud and perform a sync to the local database.

3.1.2. Malicious software metadata storage

In terms of the digests of malicious software, we could essentially use the currently available malware databases like the android malware samples [10] database, or perform cross check with Euphony[11] and the AndroZoo application dataset [12] to label the malwares across the whole Platform, then perform the proposed SHA-256 hashing function and store the Results on the cloud, from which the database performs its sync.

3.1.3. Graphical and textual spam meta-data storage

For malicious and chicanery pictures, the Authority would perform the Neural Hash that is proposed above in Section 2.2.3, then the hashing value is distributed from the cloud to the clients, then the clients would keep the digests in the secured database locally.

As for textual spam data, we could utilize existing spam text databases like the SpamAssassin database from Apache [13], which contains spam texts as well as ham texts. For the authorities to classify and label spam texts, a variety of machine learning techniques could be deployed. Through extensive research, Jancy Sickory Daisy & Rijuvana Begum implemented both the Nave Bayes method and the Markov Random Field, complementing the advantages of both kinds of machine learning, which could efficiently and accurately identify the spam texts [14].

The vectors of the malicious text are then synced to the local security database.

3.2. Pattern matching and classification process

As the name indicates, this part of the system focuses on recognizing the threat, which is done by pattern matching the Hash digests, judging if the system has actions that

deviate from the normal behavior of the legitimate user, as well as classifying whether the textual and graphical data that the unit gathered has malicious contents.

However, the processing cannot rely solely on the local context, in order to not just increase the accuracy of the whole Matching process, but also to flag the device if it is used by a malicious user, the processing itself will redirect some of the data to the Communication Sub Unit, which is connected to the authorities.

For privacy concerns, the PSI (Private Set Intersection) protocol is used for that the authority can only learn from the suspicious contents of the phone. Furthermore, the Threshold Secret Sharing protocol is used, for that the authority will not learn anything from the phone itself unless the suspicious and false contents on the phone exceed a certain threshold. The details of how both protocols are being used are discussed below.

3.2.1. Pattern matching algorithm

Since the computing is done real-time, the efficiency of the pattern matching algorithm is crucial to the system's functionality.

The primitive approach towards pattern matching is the BM algorithm, it utilizes a shifting array that derives from the pattern string, indicating how the pattern string is moved when a mismatch happened.

The method to compute the shift [] array:
when x doesn't appear in the pattern P or it only appears at the end of P, shift[hash(x)]=m; where hash(chr[i]) is the function that projects different characters into the array index of shift[], the m is the length of the pattern P.

Other circumstances: shift[hash(x)]=m-j, where j=max{ j | P[j]=x, 1≤ j≤m-1).

Here is an instance:

```
Text:   acddefebfbfe
Pattern:bfe
d and e is a mismatch, the shift[hash(d)]=3, so the pattern shifts right for 3 characters.
text:   acddefebfbfe
pattern:    bfe
```

To optimize the BM algorithm, Horspool came up with the BMH algorithm.

The Boyer-Moore-Horspool algorithm compares characters from the end of the pattern to its beginning. When characters do not match, searching jumps to the next matching position in the pattern [15].

The skip [] array or the shift [] array is computed as:

```
int *GetSkip(char *p, int pLen)
        {
            int i;
            *skip=(int*) calloc(|Σ|, sizeof(int));
            for(i=0;i<|Σ|;i++)  skip[i]=pLen;
            for(i=0;i<pLen-1;i++)   skip[p[i]]=pLen-i-1;
            return skip; /* |Σ| is the number of entries of the distinct characters in the string,
            for example, if p is composed solely from letters, the |Σ| would be 26.*/
            }
```

The matching algorithm can be described as:

```
int BmhSearch(char *t, int tLen, char *p, int pLen, int *skip)
        {/*t points to the target string that contains the pattern string, p points to the pattern*/
        int i=0, j;
        while (i<=tLen-pLen)
            { for(j=pLen-1; j>=0 &&p[j]==t[j+i];j--);
            If(j<0)
            return 1;
            i+=skip[t[i]];
            }
        return -1;
        }
```

In terms of real-world applications, the BMH could achieve a Time complexity of O(m+n), which is a big step up from the BM algorithm.

The two algorithms above can only function with one pattern string, although they can match the string immediately, since the M-ISDS system has multiple sources of the pattern to be matched, we introduce the AC_MBM2 algorithm to the system, [16] which matches multiple sources of pattern at once.

Assume that the search target string is T, the length of T is n; The array P{P[0],P[1],…,P[s-1]} to represent the many patterns, the array that indicates the length of the pattern strings are {len[0],len[1],len[2],…,len[s-1]}, m is the length of the shortest pattern string, the character set is the array{chr[0],chr[1],…,chr[|Σ|]}, where |Σ| is the number of entries of the distinct characters in the character set.

The algorithm aims to form a tree from the multiple patterns, then makes the overlapping suffix the root of the tree, the tree is called the pattern tree.

For example, the multiple patterns: time, tiring, tinted, tired, tinsel forms the pattern tree as Fig.5 below

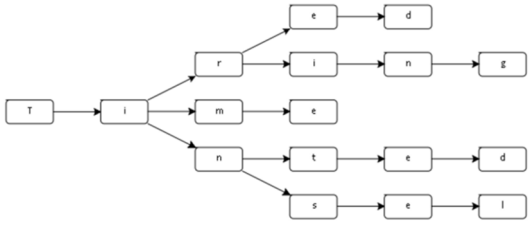

Figure.5 The pattern(rule) tree

In total, |Σ| of the pattern trees are formed and then the skip array for each of the pattern is computed as:

```
For(i = 0; i < | Σ | ; i++)
M[i] = the length of the shortest pattern in the i th pattern tree;
For(int i = 0; i < | Σ | ; i++) {
  For(int j = 0; j < | Σ | ; j++)
  // iterate over all the substrings that have a length of 2
  For(k = 0; k < | Σ | ; k++)
  //initialize the array
  skip[hash(i,  chr[j],  chr[k])] = m[i];
}
for (i = 0;  i < s;  i++) {
    //iterate over all the pattern strings, whereas S is the number of the pattern strings
  int n = len[i]; // len[i]is the length of the pattern P[i]
  for (j = n - 1; j >= 1; j--) { /*compute the distance of the leftmost part
  of the substring P[i][j-1]P[i][j] to the leftmost part
  of the pattern string(the value of skip)*/
    If((j - 1) < skip[hash(P[i][0], P[i][j - 1], P[i][j])]);
    // P[i][0] determines the offset of the certain pattern tree
  }
```

The matching process is that the pattern tree moves from rightmost to the left, when moving, the whole tree moves instead of only a single pattern; the matching of the characters are from the root of the pattern tree to the leaf of the tree.

The steps can be broken down into the following:

Align the root to the end of the target which is shifted left L characters, where L is the length of the shortest pattern in the rule tree.

Start matching from the root of the tree, if the pattern tree doesn't contain the substring chr[i-1]chr[i] which belongs to the target string, the tree is shifted left L times.

If chr[i-1]chr[i] exists in the pattern tree, move the tree left so that the chr[i-1]chr[i]in the target aligns the chr[i-1]chr[i] in the pattern string, the shifting distance is determined by the skip array.

Here is an example of the algorithm, the target string is 'timeisonmyside', the pattern strings are 'time' 'tiring' 'tired' 'tinsel' 'tinted', the pattern tree is mapped as the Fig.5 above, L=4. The first comparison involves the target string shifting left 4 times and comparing with the pattern tree: 'timeisonmyside', since s≠t, and 'ys' doesn't exist in the whole pattern tree, the shifting distance is equal to L, which is 4. The next comparison: 'timeisonmyside', then the character o is compared to the root of the pattern tree.

The above process is then repeated until a match happens or the search is complete.

In this article, 100MB of data from an existing commercial site is used and the 'time' command is used to test the 3 above algorithms. The standard Snort Dataset is used as the rules, the results are as follows.

Table 1: Time used in each of the pattern matching algorithms

Algorithms	BM	BMH	AC_MBM
Rules used			
68	7.9	6.5	5.02
268	10.46	8.9	5.52
468	24.59	20.3	5.44
668	35.92	32.4	2.58
786	40.34	38.5	5.78

The results show that as rules and patterns increase, multi-pattern matching algorithms are essential for the efficiency and the availability of the system. As a result, the Decision support system utilizes AC_MBM pattern matching, which boosts the performance of the system by quite a huge margin.

3.2.2. Call and SMS classification

The caller and the sender's numbers will perform a pattern matching in the database of the flagged suspicious Callers and SMS senders, if there is a match, the call will be redirected to the local self-contained processing unit, where the call or the SMS will be recorded and logged, then they would be kept in a secure database inside the unit for a certain amount of time, say, for 15days, which is more than enough for evidence collection when needed.

3.2.3. Malicious application digest pattern matching

Once an application is downloaded or executed, the digest of the application will be computed and sent to the decision support unit, where pattern matching is performed on the digest to the known malicious software digests, if there is a match, the application and its affiliated processes will be terminated, then the application will be kept inside a local sandbox in the local self-contained unit, where the user has the final say to whether keep the application as it is, provided that the alert has been sounded, or to delete the application and purge its affiliating data.

3.2.4. Graphical data processing

The neural hash of the images is sent to the Decision support system, where pattern matching is performed against the known malicious pictures hash list. In order to prevent inference attacks on the database and preserve the users privacy, the matching will use PSI protocols.

The goal of PSI protocol is to not reveal the matching result to the user, and that the server will do the final decision on whether the picture is deceitful or not.

Before establishing connections with the server, the neural hash of the pictures will be looked up against a blinded hash table database which is filled with dummy entries, the neural hash value will point to an entry, which is then used to encrypt the picture itself as a payload to the neural hash.

Then the neural hash and the encrypted picture is sent to the authority's server, since the server shares the same blinded hash table database with the user's device, except that the server only has the entries corresponding to known malicious pictures, the server uses the same lookup process. If there's a match in the database, the server can decrypt the payload data, if there isn't, it indicates that the matching in the user's device is false, the server does not have the key to decrypt the payload, which ensures that the server will not learn anything from the payload data, and that the possible malicious user will not learn anything on the local known malicious pictures hash database.

Moreover, to reveal the possible malicious users, the threshold secret sharing protocol is used. Since this protocol enables a secret to be split into distinct shares so the secret can then only be reconstructed from a predefined number of shares (the threshold), if the server has numerous matches from one single user, exceeding the threshold, the server could reconstruct a shared secret key to the user's device, then additional data inside the local processing unit will be sent to the communication unit and sent to the authority for further classifications and possible evidence collection.

3.2.5. Textual data classification and processing

The word vectors will be passed to the Pattern matching and classification unit, where it will perform cross-check between predefined malicious text vectors in the local database and local machine learning algorithms with the classifiers trained by the server from the authorities.

While the predefined text will be flagged, the ones labeled by the machine learning algorithm will be sent to the Communication sub unit and queried against the server.

The system utilizes the multistage detection framework for spam detection proposed by Bo Feng, Qiang Fu, Mianxiong Dong, Dong Guo, and Qiang Li [17], which does its primitive labeling process on local with predefined classifiers and enables further classification and detection of the spam texts with CNN model on the server.

The results will be returned and logged by the server, then the user device will send out alerts according to the final results.

The process has a high accuracy while remaining almost no compromise in computational power on the local device.

3.2.6. Anomaly data processing and classification

The logging data is used here to classify potentially unauthorized actions and malicious occurrences in user behaviors.

To achieve this target, the system builds the behavioral profile through the file logs, internet activities and other activity logs gathered in the Anomaly Detection sub unit.

The system utilizes supervised learning, using the Bayesian networks along with the cross-evaluation of holdout split method proposed in the issue of Evaluation of anomaly-based IDS for mobile devices using machine learning classifiers [18], which could effectively send out alerts when anomaly activities are logged, will remain responsive and relatively fast in terms of processing time.

4. CONCLUSION

The proposed hybrid system based on a series of mechanisms to monitor and filter most kinds of cyberattacks there is, while remains its catering to user's privacy, effectively detecting threats and carrying out the corresponding measurements to contain or alert the threats to the end user, even when the user is off of internet connection.

However, the whole system still takes a toll in terms of device processing power, and it has high requirements for third-party authorities, which severely limits the system.

Due to the scale of the whole system involves not only a huge chunk of processing, but also the co-operations with authorities and cloud computing, the system is not yet realized so real-life tests can only be performed on the per-unit scale. Furthermore, Controversy on Ethics is not thoroughly looked into in this work, which could be a limiting factor when the system is widely implemented.

REFERENCES

1. Janis Griffin (2021) What Is an Intrusion Detection System (IDS)? https://logicalread.com/intrusion-detection-system/
2. Wang, X., Yu, H. (2005). How to Break MD5 and Other Hash Functions. In: Cramer, R. (eds) Advances in Cryptology – EUROCRYPT 2005. EUROCRYPT 2005. Lecture Notes in Computer Science, vol 3494. Springer, Berlin, Heidelberg. https://doi.org/10.1007/11426639_2
3. Xiaoyun Wang, Hongbo Yu, & Yiqun Lisa Yin (2005). Efficient Collision Search Attacks on SHA-0. In In Crypto (pp. 1–16). Springer-Verlag.
4. Sharfah Ratibah Tuan Mat, Mohd Faizal Ab Razak, Mohd Nizam Mohmad Kahar, Juliza Mohamad Arif, & Ahmad Firdaus (2021). A Bayesian probability model for Android malware detection. ICT Express.
5. APPLE INC. (2021) CSAM Detection Technical Summary. https://www.apple.com/child-safety/pdf/CSAM_Detection_Technical_Summary.pdf
6. Sakaguchi, K., Duh, K., Post, M., & Van Durme, B.. (2016). Robsut Wrod Reocginiton via semi-Character Recurrent Neural Network.
7. Rong, X.. (2014). word2vec Parameter Learning Explained.
8. Dataset for scam callers recognition and classification http://www.scdata.net.cn/common/cmpt/%E8%AF%88%E9%AA%97%E7%94%B5%E8%AF%9D%E8%AF%86%E5%88%AB_%E6%8E%92%E8%A1%8C%E6%A6%9C.html
9. www.geeksforgeeks.org. Residual Networks (ResNet) – Deep Learning https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/
10. Ashishb. Collection of android malware samples https://github.com/ashishb/android-malware
11. Hurier, M., Suarez-Tangil, G., Dash, S., Bissyandé, T., Le Traon, Y., Klein, J., & Cavallaro, L. (2017). Euphony: Harmonious Unification of Cacophonous Anti-Virus Vendor Labels for Android Malware. In 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR) (pp. 425-435).
12. Allix, K., Bissyandé, T., Klein, J., & Le Traon, Y. (2016). AndroZoo: Collecting Millions of Android Apps for the Research Community. In Proceedings of the 13th International Conference on Mining Software Repositories (pp. 468–471). ACM.
13. Apache Software Foundation. (2021) Apache SpamAssassin Spam Filter https://spamassassin.apache.org/index.html
14. Kaddoura S, Chandrasekaran G, Elena Popescu D, Duraisamy JH. A systematic literature review on spam content detection and classification. PeerJ. Computer Science. 2022; 8: e830. DOI: 10.7717/peerj-cs.830. PMID: 35174265; PMCID: PMC8802784.
15. LianYing Min, & TingTing Zhao (2006). Research and improvements on pattern matching algorithms. Computers and Modernizations (8), 4.
16. Mike Fisk, & George Varghese (2001). Fast Content-Based Packet Handling for Intrusion Detection [White paper].
17. Feng, B., Fu, Q., Dong, M., Guo, D., & Li, Q. (2018). Multistage and Elastic Spam Detection in Mobile Social Networks through Deep Learning. IEEE Network, 32(4), 15-21.
18. Damopoulos, D., Menesidou, S., Kambourakis, G., Papadaki, M., Clarke, N., & Gritzalis, S. (2012). Evaluation of anomaly-based IDS for mobile devices using machine learning classifiers. Security and Communication Networks, 5, 3-14.