A Honey pot Implementation for security Enhancement in IOT System using AES and Key management

Pothumani.Parvathi[a], Anuradha Chinta[b], S.R.Chandra Murthy Patnala[c]

[a]Research Scholar, Computer Science and Engineering, University College of Engineering and Technology, Acharya Nagarjuna University, ANUCET, Guntur, A.P. India. Email: parvathi047@gmail.com
[b] Assistant Professor, Department of Computer Science and Engineering, V.R.Siddhartha Engineering College, Vijayawada, A.P. India.
[c] Assistant Professor, Department of Computer Science and Engineering, University College of Engineering and Technology, Acharya Nagarjuna University, ANUCET, Guntur, A.P. India.

Article History: Received: 10 November 2020; Revised 12 January 2021 Accepted: 27 January 2021; Published online: 5 April 2021

Abstract: Honey pot technology is a trap technology in network defence. It studies and learns the adversary's attacks by attracting and deceiving attackers and recording their attack behaviours. Target and attack methods to protect real service resources. However, traditional honey pot technology has inherent shortcomings such as static configuration and fixed deployment, and it is very easy to be attacked. Attacker identification bypasses and loses decoy value. Therefore, how to improve the dynamic and deceptive nature of honey pots has become a key issue in the field of honey pots. This paper is the research results in the field of honey pots at home and abroad are sorted out. First, the development history of honey pots is summarized. Then, the key technology of honey pots is the core, and the implementation process, department. It analyzes the method of signature, anti-identification thinking, and the theoretical basis of game; finally, it classifies and narrates the defence results of different honey pots in recent years, and develops the honey pot technology. Development trends are analyzed and stated, aiming at potential security threats, and looking forward to defence applications in emerging areas. In this paper, novel AES algorithm has been developed. In addition key management feature is added which provides more secure for registered users..

Keywords: Honey pot, Attacks, AES, Key management, Password based key derivation function

1. Introduction

The requirement of all the information and machines from hackers or attackers appears to connect various computer networks and systems. Applying the merely soil or modifying valuable data or very own gain would help gain a few confidential facts [1]. Towards information structures, a massive form of attack activities has been increased in parallel to the speedy technological trends. Information security has become greater essential due to the higher crime rates of informatics in technology [2]. It is required to design an efficient system that should prevent, detect, and block intrusion for significantly less fake high-quality price. A system's security is not ensured, although there are various safety features for defending the organization's computer assets or a home user. Designing a secure system is challenging, and many costs should have to be spent on system development and design [3]. Some network assaults with most occurring kinds include denial of service attacks, password-based attacks, identity spoofing, data amendment, and eavesdropping.

The installation of an intrusion detection system is required for a business enterprise to overcome many types of assaults that help defend the confidential statistics exchanged over the network. For the threats analysis and alert the administrator, an Intrusion Detection System (IDS) has monitored the tactics on a network or a device [4]. All network traffic can observe with the assistance of IDS. The malicious traffic detection and decode and log some particular packets at a centralized point is an easier task for a honey net. The burden of IDS can reduce significantly by honey pot technology, which is the strong complement to IDS. To provide the tracking of attack source, the attacker can access the information with the most significant degree [5]. Intrusion Detection and Prevention Systems (IDPS) 's essential tasks are the feasible incidents' detection and log facts maintenance and log reporting in attempts. Similarly, the organizations have utilized IDPS for various activities like deterring people from violating protection guidelines, documenting contemporary threats, and detecting problems with protection rules. For the safety infrastructure of nearly every employer, a crucial addition has made by IDPS [6].

2. Related Work

In [7], the author demonstrates the vital role and utilization of Honeypot technologies to detect, identify, and collect data about various network threats. The continuous evolution of Honeypots is made due to its broad potential. In different environments, technologies can be implemented as they have remarkable benefits. The false positives can reduce by them while an extremely flexible tool provides for customizing easily in different threats and environments. By using Honeypots, the addressing of common external and internal threats have been made. Honeypots' contribution is to the early indication and confirming the advanced insider threats by integrating the capabilities of Honeynets and Honeytokens.

In [8], the authors have proposed a system of shadow Honeypot based intrusion detection, which can be used to gather the network's intrusion. Shadow Honeypot is integrated with the intrusion detection system for evaluating the performance of detection. The specific kinds of failure can detect using the shadow to detect the potential attacks. Both misuse detection and anomaly integrate with honeypot to overcome the IDS deficiencies. For verifying whether the packet is malicious or not, the IDS packet is collected by the shadow Honeypot. For processing, the rerouting of data towards the shadow version is done or else routed to the destination application when the data is infected. The malicious packet effects all kinds of state changes, and they rolled back to its safe state. By detecting better intrusion and reducing the rate of false positives, the overall security of a system may improve by the proposed system.

In [9], the authors have defined the Honeypot purpose for diverting the intruders away from essential sources and considering the techniques of an attacker. To create honeypots, one of the maximum used tools is honeyd which can generate a large number of logs in case of dense attack traffic, leading to more consumption of disk space in case of dense attack traffic. During the analysis and processing of enormous log, the resource consumption and massive time are the drawbacks. By proposing two essential modules, all these disadvantages can be reduced. Here, the packets are collected in the first module in the network and the second module is used to analyse the gathered packets for generating the summarized information of graphs and captured packets for the administrators of security. This application also monitors the packet data related to web traffic. The required space for log file reduces significantly based on the analyzation of experimental results and the reports generated dynamically according to the user needs.

In [10], the authors have proposed a system NIDS-SA or Network Intrusion Detection System Snooper Agent, which includes combining the statistical anomaly detection approach and the rule-based detection algorithm. Three necessary components like Snooper Agent (SA), Intrusion Detection Coordinator (IDC), and Intrusion Detection Node (IDN) include in the proposed method for supporting the capability of active monitoring. To process the intrusion inferring, intrusion detection, and attacking Snooper, these three components are operated in a synchronized way. Using the first component IDN, the packets are captured, demultiplexed them, detect local intrusion, and infer intrusion. For managing and communicating the IDNs, the second component installs in a workstation of administration. For a collection of information, different snoop functions include in RA. The statistical inference and pattern matching are also included in NIDS-SA. For achieving the ability to secure communication between IDNs and IDC, the proposed NIDS-SA implements the cryptography-based mechanisms additionally.

In [11], the author mentions the concept of coordinating Honeypot and IDS, which can form and set off the snort rule according to the view of data sending using Honeypot server. Further, the IDS will analyse the collected information from Honeypot, and the rules generate automatically. For filtration of packets send by the user, these rules have become active on the network. In a snort system, the comparison of automatically generated rules is made with default rule for a similar pattern. By computing the IDS server effectiveness from the attacking, the proposed technique shows better performance.

In [12], the author proposes a method for preventing the assaults by deploying the nodes of intrusion detection in MANETs. Two kinds of assaults like black hole attack and wormhole attacks are addressed in the paper. For restricting the attacks of black hole and wormhole, the modules known as AntiBlackhole and AntiWorm have been used respectively. The IDS nodes can make the successful detection and mitigation of malicious nodes through the experimental results. By making alterations to the algorithms, the paper discusses proposing an IDS that should be capable of identifying both black hole and wormhole. To fight against corresponding attacks, the same tables can share by the proposed modules AntiBlackhole and AntiWorm. The necessary information for the detection of both attacks can keep by the regular nodes and IDS nodes.

3. Proposed method

To reduce vulnerability for brute-force attacks, PBKDF1 (Password-Based Key Derivation Function 1) and PBKDF2 are used with a sliding computational cost in cryptography.

In the series of RSA Laboratories' Public-Key Cryptography Standards (PKCS), PBKDF2 is a part specifically in PKCS #5 v2.0, also published as Internet Engineering Task Force's RFC 2898. It has succeeded than PBKDF1 which could generate derived keys only up to 160 bits long. According to the publication of RFC 8018 (PKCS #5 v2.1) in 2017, PBKDF2 is the recommended option for password hashing.

a. Purpose and Operation

A pseudorandom function implements in PBKDF2 for the input password or passphrase like hash-based message authentication code (HMAC) in addition to the salt value. The repetition of the process is performed frequently for generating a derived key. In subsequent operations, it can be used as a cryptographic key. The password cracking is more difficult by the added computational work and is called as essential stretching.

The minimum number of iterations is recommended as 1000, but it is increased with the increased CPU speed when the standard was formulated in the year 2000. However, 4096 iterations recommended for a Kerberos standard in 2005, 2000 are used for iOS 3 for Apple, 10000 for iOS 4; 5000 iterations are used for JavaScript clients in LastPass in 2011 and 100000 iterations for server-side hashing.

The ability to use precomputed hashes (rainbow tables) is reduced for attacks by adding salt to the password. That means the individual testing of multiple passwords has to be added not all at once. At least 64 bits of the last length suggests by the standard. However, 128 bits of a salt length is recommended by the US National Institute of Standards and Technology.

b. Key Derivation Process

Five input parameters contain in the key derivation function of PBKDF2:

DK = PBKDF2 (PRF, Password, Salt, C, dkLen)

Where

- DK is the generated derived key
- dkLen indicates the desired bit-length of the derived key
- Salt is a sequence of bits, i.e., a cryptographic salt
- c refers to the desired number of iterations
- Password relates to the master password which produces a derived key
- A pseudorandom function of two parameters with output length hLen (e.g., a keyed MAC) is known as PRF

Each derived key DK's hLen-bit block T_i is determined by using below equation.

$DK = T_1 + T_2 + \dots + T_{dklen/hlen}$

T_i = F (Password, Salt, c, i)

The chained PRFs with the xor (^) operations of c iterations considers as the function F. As the Salt, the password uses by the PRF's first iteration and the encoding of PRF key concatenated with i is made as a big-endian 32-bit integer like the input. Here, i indicates as a 1-based index. As the PRF key, the password utilizes by the PRF subsequent iterations and the previous computation output of PRF as the input:

$$\mathrm{F}\ (\mathrm{Password, Salt, c, i}) = U_1 \wedge U_2 \wedge \dots . U_c$$

Where

U_1 = PRF (Password, Salt, INT_32_BE (i))

U_2 = PRF (Password, U_1)

U_C = PRF (Password, U_{c-1})

For example, VPA2 uses:

DK = PBKDF2 (HMAC-SHA1, passphrase, ssid, 4096, 256)

In PBKDF1, a simpler process includes: the initial U forms based on PRF (Password + Salt), and the following ones PRF (Uprevious). The extraction of key is done as the final hash's first dkLen bits, that's the reason a size limit is there.

c. The Advanced Encryption Standard (AES)

For both encryption and decryption, the same key uses by a symmetric block cipher, i.e., the Advanced Encryption Standard (AES) which is different from DES in various ways. A variety of block and key sizes allow by the algorithm of Rijndael and not just the 56 and 64 bits of the block and key sizes of DES. From 128, 160, 192, 224, 256 bits, the block and key can be chosen independently, and there is no necessity of being the same. A block size of 128 bits and a choice of three keys – 256, 192, and 128 bits can accept only by the algorithm based on the AES standard states. The common name is altered to AES-256, AES-192, and AES-128 respectively by relying on which version is utilized. AES is varied from DES as it is not a Feistel structure. The half of the data block modifies the other remaining half of the data block, and the haves is swapped by recalling in a Feistel structure. By using permutations and substitutions, the parallel processing of overall data block is done during each round.

The dependency on several AES parameters is on the key length. The number of rounds is 10 when the key size is 128. However, it is 12 and 14 for 192 and 256 bits respectively. The 128-bit key is utilized as the most common key size currently. The particular implementation describes the AES algorithm. The overall structure of AES can be seen in figure1.

Inner Workings of a Round

In the algorithm, the initial stage is an Add round key followed by nine rounds of four stages and the tenth round of three stages. For both encryption and decryption, the algorithm implements with the exclusion of each stage of a round. The inverse of the decryption algorithm is the counterpart to the encryption algorithm. Accordingly, the four stages involve as mentioned below:

1. Substitute bytes
2. Shift rows
3. Mix Columns
4. Add Round Key

The stage of Mix Columns leaves out in the tenth round. The following stages include in the first nine rounds of the decryption algorithm:

1. Inverse Shift rows
2. Inverse Substitute bytes
3. Inverse Add Round Key
4. Inverse Mix Columns

The stage of Inverse Mix Columns leaves out in the tenth round. Each of these phases will consider in detail.

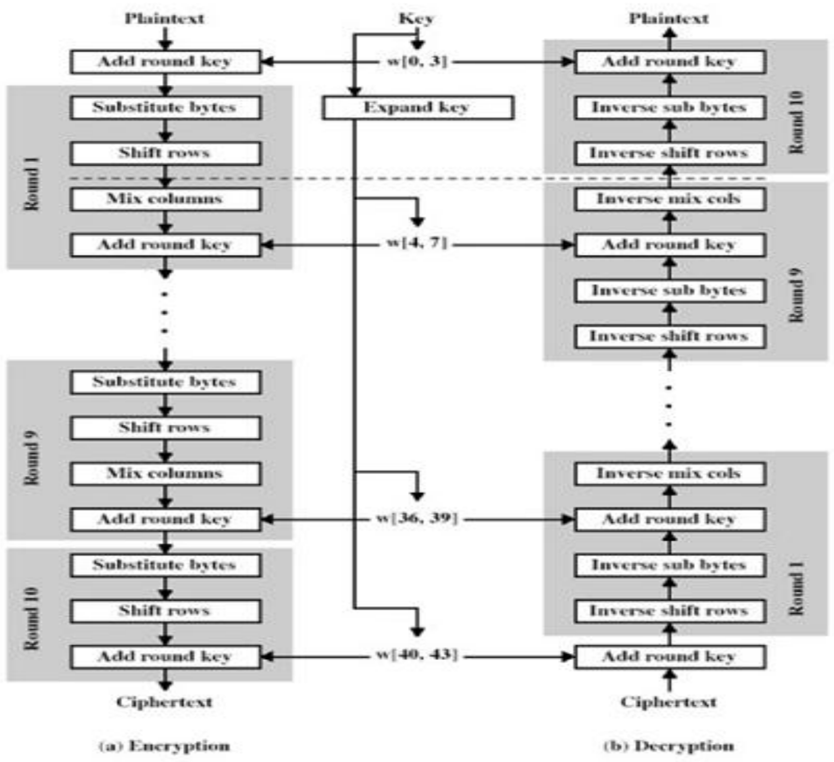


Figure 1: Overall structure of the AES algorithm

Substitute Bytes

With a 16×16 matrix of byte values, a table lookup includes in this stage (known as Sub Bytes) i.e., an s-box. An 8-bit sequence with all possible combinations$2^8 = 16 \times 16 = 256$include in the matrix. For creating the tables of s-box, a well-defined method is used as it is not just a random permutation of these values. This process is shown by Rijndael designers that is not same as the s-boxes in DES for which rationale was not given. How the s-boxes are made up will not be concerned and can take them as table lookups simply.

Shift Row Transformation

In the figure 2, this stage i.e., ShiftRows is shown and it is a simple permutation. The stage is operated as follows:

- The state on the first row is not changed.
- The second row shifts to the left by 1 byte circularly.
- The third row shifts to the left by 2 bytes circularly.
- The fourth row shifts to the left by 3 bytes circularly.

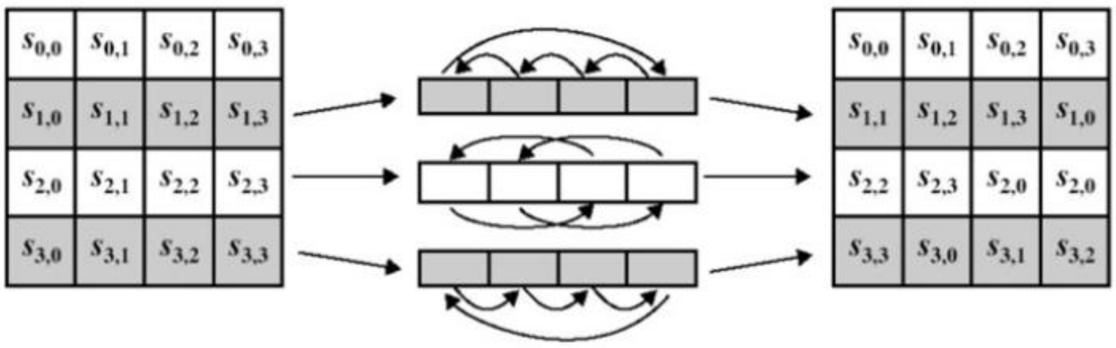

$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,1}$ $s_{1,2}$ $s_{1,3}$ $s_{1,0}$
$s_{2,2}$ $s_{2,3}$ $s_{2,0}$ $s_{2,0}$
$s_{3,3}$ $s_{3,0}$ $s_{3,1}$ $s_{3,2}$

Figure 2: ShiftRows Stage

Mix Column Transformation

A substitution includes in this stage called MixColumn, but the arithmetic of GF(2^8) is used. Individually, each column operates. In a column, a new value is mapped to each byte known as a function of all four bytes in the column. The following matrix multiplication on the state can evaluate by the transformation (see below figure 3):

$$\begin{bmatrix} 02 & 0301 & 01 \\ 01 & 0203 & 01 \\ 01 & 0102 & 03 \\ 03 & 0101 & 02 \end{bmatrix} \begin{bmatrix} S_{0,0} & S_{0,1}S_{0,2} & S_{0,3} \\ S_{1,0} & S_{1,1}S_{1,2} & S_{1,3} \\ S_{2,0} & S_{2,1}S_{2,2} & S_{2,3} \\ S_{3,0} & S_{3,1}S_{3,2} & S_{3,3} \end{bmatrix} = \begin{bmatrix} S'_{0,0} & S'_{0,1}S'_{0,2} & S'_{0,3} \\ S'_{1,0} & S'_{1,1}S'_{1,2} & S'_{1,3} \\ S'_{2,0} & S'_{2,1}S'_{2,2} & S'_{2,3} \\ S'_{3,0} & S'_{3,1}S'_{3,2} & S'_{3,3} \end{bmatrix} \qquad (7.1)$$

$$S'_{0,j} = (2\, S_{0,j}) \oplus (3 \bullet S_{1,j}) \oplus S_{2,j} \oplus S_{3,j}$$

$$S'_{1,j} = S_{0,j} \oplus (2 \bullet S_{1,j}) \oplus (3 \bullet S_{2,j}) \oplus S_{3,j}$$

$$S'_{2,j} = S_{0,j} \oplus S_{1,j} \oplus (2 \bullet S_{2,j}) \oplus (3 \bullet S_{3,j})$$

$$S'_{3,j} = (3 \bullet S_{0,j}) \oplus S_{1,j} \oplus S_{2,j} \oplus (2 \bullet S_{3,j})$$

Where $\bullet$ indicates multiplication over the finite field GF (2^8).

The definition of product matrix each element is the sum of elements products of one row and one column. In GF (2^8), the individual multiplications and additions are processed.

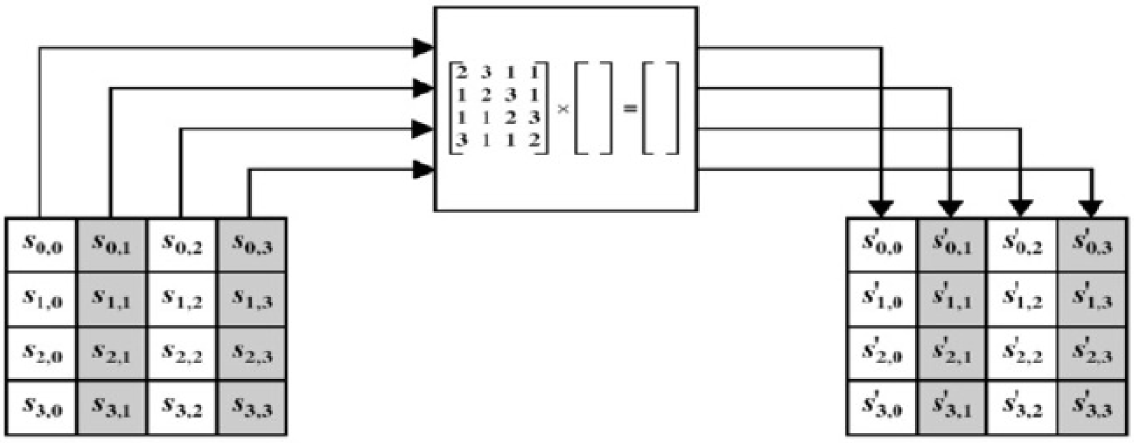

2 3 1 1
1 2 3 1
1 1 2 3
3 1 1 2
×
=
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$s'_{0,0}$ $s'_{0,1}$ $s'_{0,2}$ $s'_{0,3}$
$s'_{1,0}$ $s'_{1,1}$ $s'_{1,2}$ $s'_{1,3}$
$s'_{2,0}$ $s'_{2,1}$ $s'_{2,2}$ $s'_{2,3}$
$s'_{3,0}$ $s'_{3,1}$ $s'_{3,2}$ $s'_{3,3}$

Figure 3: Mix columns stage

Add Round Key Transformation

The bitwise XORed with the 128 bits of the round key is the 128 bits of state in this stage, i.e. AddRoundKey. Between the one word of round key and a state column's 4 bytes, the operation can view as a column wise operation. The efficiency could be achieved and every bit of state also affected by this transformation which is as simple as possible.

Equivalent Inverse Cipher

The ciphers of decryption are not similar to the ciphers of encryption. For both these encryption and decryption ciphers, the form of the key schedule is similar. The applications that require encryption and decryption, two different software or firmware modules, are the main drawback. In the case of implementation, decryption is less efficient slightly. For below two reasons, encryption was essential compared to the decryption:

1. The encryption is only used for the cipher mode of CFB and OFB.
2. For constructing a message authentication code, AES can utilize with any block cipher and encryption only use for this.

4. Results and Discussion

In this paper we have added key management features to secure registered genuine users account details and for each user we will create keys with an algorithm called PBKDF2 (Password-Based Key Derivation Function 2) and using this KEYS we will encrypt users account details by using AES algorithm. By encrypting user account, we can avoid misuse of data as internal employees of database can view all those records if we store in plain format and can misuse it and by encrypting, we can avoid such misuse. Another advantage of encrypting data with keys is we can send encrypted data in network so no hackers can understand it after stealing from network. In this work user data will exchange between browser and honeypot in the form of encrypted data.

In below showing how we are generating keys and performing encryption

First, we generating the key with PBKDF2 for AES,

$Password = \text{“s3cr3t} * \text{c0d3”}$

$\text{PasswordSalt} = \text{“76895"}$

$\text{key} = \text{Pbkdf2. PBKDF2 (password, passwordSalt). Read (32)}$

Return key

Defined the encryption (plaintext): AES data encryption

$aes = pyaes. AESModeOfOperationCTR\ (getKey(), pyaes, Counter$

$(311295470350000473029547878320438873439835634678846354163632435768123543223))$

$\text{ciphertext} = \text{aes. encrypt (plaintext)}$

return ciphertext

Defined the encryption (plaintext): AES data encryption

$\text{aes} = \text{pyaes. AESModeOfOperationCTR (getKey(), pyaes, Counter}$

$(311295470350000473029547878320438873439835634678846354163632435768123543223))$

$\text{decrypted} = \text{aes. decrypt (enc)}$

return decrypted

In above format we can see code for generating keys and to perform encryption and now in below we can see user SIGNUP.

Table 1: user details

username	password	Contact no	email	address
LbwAYGo=		9652768976	kumar@gmail.com	Hyd
LKYFb2vS2g==	LKYFb2vS2g==	9652768976	johnson@gmail.com	Hyd

In above table 1 we can see username and password of johnson is store as encrypted data and no internal employees who access this data can identify correct username and password so we are protecting this application from internal (internal means employees who are accessing this data and they too can steal and misuse the data) and external (external means outsider users who try to access without username and password) attackers.

Now in we designed 5 honeypot servers and for each attacker request different honeypot servers will responded.

1. Login Request

Local host Starting Server ON 5555

2. Login Request

Local host Starting Server ON 6666

3. Login Request

Local host Starting Server ON 7777

4. Login Request

Local host Starting Server ON 8888

5. Login Request

Local host Starting Server ON 9999

In above started all 5-honeypot server and one is on port 5555 and second one is on 6666 and third one is on 7777 and fourth is on 8888 and 5th 0n 9999. Now in below we try to access IOT data as attacker

Table 2: View IoT Data

IoT ID	Sense temperature	Sense Date
7	33	2020/12/29-11:56:17
7	68	2020/12/29-11:56:37
7	46	2020/12/29-11:56:57
7	59	2020/12/29-11:57:17
7	32	2020/12/29-11:57:37
7	47	2020/12/29-11:57:57
7	24	2020/12/29-11:58:17

At honeypot server we can see details for above user request mentioned as in table 2.

Quit the server with CTRL-BREK.

“GET/ index.html HTTP/1.1” 200 2061

“GET/ Register.html HTTP/1.1” 200 3338

Record Inserted

“POST/ UserRegister.html HTTP/1.1” 200 3362

“GET/ Login.html HTTP/1.1” 200 2276

Local host

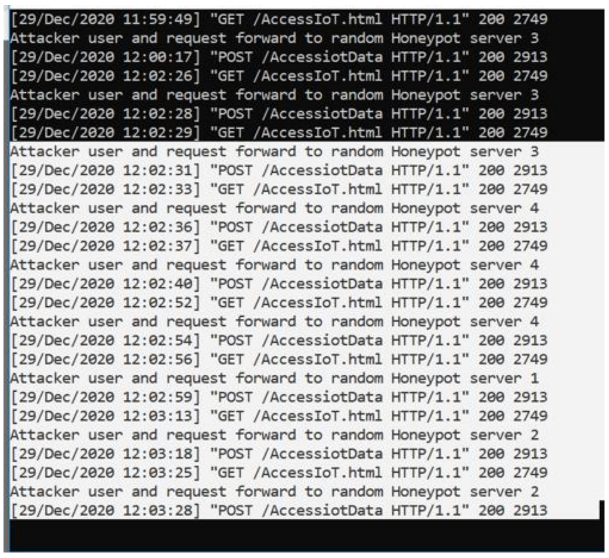
“POST/ UserLohin.html HTTP/1.1” 200 2094

“GET/ AccessIoT.html HTTP/1.1” 200 2749

Attacker user and request forward to random Honeypot server 3

“AccessiotData HTTP/1.1” 200 2913

In above honey pot server we can see attacker is identified and it request was processed by 3rd honey pot server and now in below figure send multiple attack request and then see which honey pot server will process it



```
[29/Dec/2020 11:59:49] "GET /AccessIoT.html HTTP/1.1" 200 2749
Attacker user and request forward to random Honeypot server 3
[29/Dec/2020 12:00:17] "POST /AccessiotData HTTP/1.1" 200 2913
[29/Dec/2020 12:02:26] "GET /AccessIoT.html HTTP/1.1" 200 2749
Attacker user and request forward to random Honeypot server 3
[29/Dec/2020 12:02:28] "POST /AccessiotData HTTP/1.1" 200 2913
[29/Dec/2020 12:02:29] "GET /AccessIoT.html HTTP/1.1" 200 2749
Attacker user and request forward to random Honeypot server 3
[29/Dec/2020 12:02:31] "POST /AccessiotData HTTP/1.1" 200 2913
[29/Dec/2020 12:02:33] "GET /AccessIoT.html HTTP/1.1" 200 2749
Attacker user and request forward to random Honeypot server 4
[29/Dec/2020 12:02:36] "POST /AccessiotData HTTP/1.1" 200 2913
[29/Dec/2020 12:02:37] "GET /AccessIoT.html HTTP/1.1" 200 2749
Attacker user and request forward to random Honeypot server 4
[29/Dec/2020 12:02:40] "POST /AccessiotData HTTP/1.1" 200 2913
[29/Dec/2020 12:02:52] "GET /AccessIoT.html HTTP/1.1" 200 2749
Attacker user and request forward to random Honeypot server 4
[29/Dec/2020 12:02:54] "POST /AccessiotData HTTP/1.1" 200 2913
[29/Dec/2020 12:02:56] "GET /AccessIoT.html HTTP/1.1" 200 2749
Attacker user and request forward to random Honeypot server 1
[29/Dec/2020 12:02:59] "POST /AccessiotData HTTP/1.1" 200 2913
[29/Dec/2020 12:03:13] "GET /AccessIoT.html HTTP/1.1" 200 2749
Attacker user and request forward to random Honeypot server 2
[29/Dec/2020 12:03:18] "POST /AccessiotData HTTP/1.1" 200 2913
[29/Dec/2020 12:03:25] "GET /AccessIoT.html HTTP/1.1" 200 2749
Attacker user and request forward to random Honeypot server 2
[29/Dec/2020 12:03:28] "POST /AccessiotData HTTP/1.1" 200 2913
```

Figure 4: multiple attack requests

In above figure 4 for each attacker request different random honey pot server was used and you can see in above screen sometime the request was processed by honeypot server 3 or 4 or 1 or 2. In above figure 4 for each IOT access data you can see chosen honey pot server.

5. Conclusion

This paper summarizes the research status of honey pots in information security and also it describes the development history of honey pots in stages in literature. From the overall trend of network security, honey pots are in line with information technology. The background of the times, featuring new technologies, emerging fields and honey pot technology as a basis, the development trend of honey pots is discussed. As a kind of master dynamic defence technology, honey pots will continue to be developed and updated, and will also be safe the research field pays more attention and application. The novel AES algorithm with Key management feature is Implemented and theoretical verification were analyzed

References

Baki, A. (2008). *Kuramdan uygulamaya matematik eğitimi*. Ankara: Harf Eğitim Yayıncılığı

Dias, L. P., Jés de Jesus Fiais Cerqueira, Karcius DR Assis, and Raul C. Almeida. "Using artificial neural network in intrusion detection systems to computer networks." In 2017 9th Computer Science and Electronic Engineering (CEEC), pp. 145-150. IEEE, 2017.

Diesch, Rainer, Matthias Pfaff, and Helmut Krcmar. "A comprehensive model of information security factors for decision-makers." Computers & Security 92 (2020): 101747.

Rajamanickam, Siranjeevi, Satyanarayana Vollala, Ruhul Amin, and N. Ramasubramanian. "Insider attack protection: Lightweight password-based authentication techniques using ECC." IEEE Systems Journal 14, no. 2 (2019): 1972-1983.

Borkar, Amol, Akshay Donode, and Anjali Kumari. "A survey on Intrusion Detection System (IDS) and Internal Intrusion Detection and protection system (IIDPS)." In 2017 International conference on inventive computing and informatics (ICICI), pp. 949-953. IEEE, 2017.

Xiaohui Bao et al., “Network Intrusion Detection Based on Support Vector Machine”, International Conference on Management and Service Science MASS '09, 2009, pp.1-4

Sanjay Kumar Sharma et al., “An Improved Network Intrusion Detection Technique based on k-Means Clustering via NaIve Bayes Classification”, IEEE-International Conference on Advances in Engineering, Science and Management (ICAESM -2012), 2012, pp.417-422

L. Spitzner, “Honeypots: catching the insider threat”, Proceedings of 19th Annual Computer Security Applications Conference, 2003, pp.170-179.

Navita Sharma and Gurpreet Singh, “Intrusion Detection System Using Shadow Honeypot”, International Journal of Emerging Technology and Advanced Engineering, Volume 2, No 8, 498-500, 2012.

Balaji Darapareddy and Vijayadeep Gummadi, “An Advanced Honeypot System for Efficient Capture and Analysis of Network Attack Traffic”, International Journal of Engineering Trends and Technology- vol. 3, no. 5, pp.616-621, 2012.

Bin Zeng, Lu Yao and ZhiChen Chen, “A Network Intrusion Detection System with the Snooping Agents”, International Conference on Computer Application and System Modeling (ICCASM 2010), 2010, pp.232-236

Albert Sagala “Automatic SNORT IDS Rule Generation Based on Honeypot Log”, 7th International Conference on Information Technology and Electrical Engineering (ICITEE), 2015, pp. 576- 580.

Ming-Yang Su “A Study of Deploying Intrusion Detection Systems in Mobile Ad Hoc Networks”, Proceedings of The World Congress on Engineering 2012 ,2012, pp1318- 1322.