Enhanced User Authentication Based on Dynamic Port Knocking Technique

Alaa Kamel Zidan
Information Technology Dep.
Menoufia University - Computers and Information Faculty,
Egypt.
alaa.k.zidan@gmail.com

Khaled Amin
Information Technology Dep.
Menoufia University - Computers and Information Faculty,
Egypt.
kh.amin.0.0@gmail.com

Tamer Fathy Ghanem
Information Technology Dep.
Menoufia University - Computers and Information Faculty,
Egypt.
tamer.ghanem@ci.menofia.edu.eg

***Abstract*—Port knocking is a passive authentication mechanism which aims to control firewall response using a sequence of connection attempts to its closed ports. Dynamic port knocking which varies in each session, faces many challenges which are knocking sequence synchronization between client and server, handling high load of normal requests, out of order knocks, lost knocks, knocking through NAT, and knocking attacks. In this paper, a proposed dynamic port knocking approach is provided. The proposed approach, with the help of intermediate IPS, enables client and target server to generate a unique dynamic knocking sequence based on a secured random seed. This process is executed only at first communication session. Next, client begins to authenticate himself by knocking the target service with different ports and different number of knocks each time a session is initiated. Client-Server knocking synchronization, lost knocks, and out of order knocks are considered for issuing a correct knocking. The proposed approach provides immunity against several network attacks such as DoS attack, replay attack, and brute forcing attack. Extensive simulation shows that the proposed work overcome other compared approaches in terms of response time, memory utilization, CPU utilization, and the number of provided features.**

Keywords: Security, port knocking, port scanning, authentication.

I. Introduction

Network attacks attempt to disrupt network normal operations by malfunctioning network devices and services. Gathering information techniques, like port scanning, is considered the first step in attack preparation. Port scanning is a well-known method that enables the attacker to identify the services running behind opened ports. So, there is a need to enable clients to connect services by targeting closed ports using a technique called port knocking.

Port knocking is an authentication technique which allows a specific system service to be available based on a received sequence of packets to target device ports in specific pattern. The target device does not respond until a correct sequence is received and hence, the service will be available to the requester. This mechanism is known as security through obscurity. Port knocking is used side by side with other authentication techniques as an additional layer of security.

Basic (static) port knocking [1] depends on sending fixed number of packet sequence (single port knock or multiple port knocks) to the same closed ports which is easy to implement. Basic knocking is not secure enough as it is susceptible to sniffing. On the other hand, dynamic port knocking is more secure than basic knocking as it is robust against brute force and sniffing attacks, but it is more complicated. Dynamic knocking sends a variable number of knocks with different port numbers each time a new session is initiated. Dynamic port knocking issues, like client/server knocking sequence synchronization, knock loss, out of order knocks reception, and resources consumption, are examples of challenges that face this type of knocking.

Firewall is main part of port knocking system as it enables client to pass through to target service if correct knocking sequence is received, otherwise, no response. Firewall controls passed traffic by configuring suitable rules for particular user based on the correctness of the knocking sequence. Intrusion Prevention System (IPS) may be used as a helper part to detect and prevent attack.

Synchronization between client and server on the current selected knocks is a must in dynamic port knocking. Out of order and lost knocks due to network congestion or buffer overflow affects the knocking process as the client and server must choose the same knocks from the sequence to enable client session to be accepted by the target service.

Dynamic port knocking algorithms serve security at the expense of processing overhead compared to static port knocking. Server should keep tracking of the generated knocking sequence of each client and his current knocking position in the sequence. This may affect the consumed resources and hence the ability to serve more clients.

Network attacks are another challenge in port knocking. Denial of service (DoS), replay attack, and brute force attacks are examples of these attacks. So, knocking systems should be immune to such attacks by making the service available to legitimate clients and detect/reject attacking requests.

In this paper, a proposed dynamic port knocking is presented. The knocking process is divided into three phases. The first and second phase, with the help of intermediate IPS, provide client identification and agreement between client and target service on knocking sequence generation in a secure way. These two phases are executed only once at the first client-server communication. Sequence generation process is selected to be simple and immune to attacks. In the third phase, user knocks the target server with selected knocks from the sequence to initiate authentication at the start of the communication session. Each initiated session has its own knocks count and knocking ports. Statistics collected by the server is fed back to IPS for client behavior evaluation. This cooperation between IPS and server enable the whole system to be robust against malicious behavior.

This paper is organized as follows: In section (II), related work is introduced. In section (III), the proposed approach is presented. Section (IV) presents results evaluation and analysis, Section (V) presents conclusion and future work.

II. Related work:

Port knocking is one of passive authentication techniques which server doesn't have any previous information about valid users. Server depends on validating knocks whoever user identification is. Port knocking validation depends on packet structure and the anonymity of target service. Port knocking processing may be accomplished by target server, or external authentication service.

In [2], a port knocking technique is presented which depends on cheating port scanners using trap server (honeypot) and fake opened ports (honeyport), while real server identity is presented as non-important peripheral. Legitimate users connect to provided services through a filtering/authentication service using a single encrypted sniffer packet that includes information about client and target service. This technique has many issues that should be considered. Filtering service is not immune against denial of service attack (DoS). User authentication at each initiated connection is another issue that increases the total system processing overhead.

Another port knocking approach is presented In [3]. This approach is based on distributing identification token (IDT) to clients with the help of domain DNS service. Before connecting target service, client connects to the DNS service to resolve the domain name and receives a DNS response that includes the IDT which is valid for 120 seconds. IDT is used by the client as a preshared key for calculating a single port knock included in the sent authentication packet to the target service. This approach is immune to information gathering related to IP-based network scanning as no response from target server is received until valid authentication is held. This approach is subject to man in middle attack as an attacker can replay a legitimate user authentication packet within the DNS expiration period to have access to the target service.

A hybrid port knocking approach is introduced In [4]. Both OAuth authentication technique and port knocking are used for gaining access to mobile web services. Steganography is used to hide the sent multiple port knocks information inside a segmented image to the target server. Firewall along with authentication server are used to validate the received requests. Symmetric and asymmetric encryption are available for requested connections which enable the system to be immune against replay attack. DoS attack is also considered as a threshold to limit the number of received packets from each client within a defined time window. Used images are limited to five images which make the system vulnerable to man in middle attack. CPU overhead is another issue due to the use of image processing which affects system response time and maximum number of concurrent users.

Another research work is presented in [5]. AES is used to encrypt knock packet between client and target server. Information like target service port number, X509 certificate, timestamp, and client IP are included in a single knock packet over UDP connection. Diffie Helman algorithm is used to distribute keys between clients and the target server. DoS attack is not considered in this research.

Despite of its higher level of security compared to fixed packet knocking, dynamic port knocking is challenging due to packet loss and out of order reception. Alternatively, fixed packet knocking techniques may integrate with external authentication services for the sake of enhancing security level which results in more processing overhead and increasing in response time. In this work, a dynamic port knocking is proposed. The knocking process, with the help of external IPS, securely provide client identification and knocking sequence generation only once at the first client-server communication on a service basis. Sequence generation process is selected to be simple and immune to attacks. As a result, processing overhead and response time are minimized in next communication sessions. At each communication session initialization, user knocks the target server with selected knocks from the sequence which varies in count and port numbers based on predefined rules to initiate authentication. Client-server knocks selection synchronization, knock loss, and out of order knocks are considered. Statistics collected by the server is fed back to IPS to evaluate client behavior. This cooperation between IPS and target service enables the whole system to be robust against many network attacks like DoS, replay attack and brute force attack.

III. Proposed work:

In this research, a proposed dynamic port knocking technique is presented. It is based on generating knocking sequence to be used between one client and one target service. Each generated knocking sequence includes several knocking blocks. Each knocking block has a variable number of knocks with different port numbers. The proposed work takes care of knocking synchronization between client and server against lost and out of order knock packets. Immunity against attacks, like brute force, DoS, and replay attacks are considered.

The proposed technique assumes that a separated IPS device and basic target server firewall are available. IPS is used to handle early service requests from clients and to be notified about some malicious knocking characteristics. Basic firewall, which included in the operating system of the target server, is configured to accept knocks without response until a correct knocking block is received, then the target service will be available to the client. Target service name and a corresponding service seed are assumed to be known for valid users which restricts the number of accepted requests to initialize a new knocking sequence.

The proposed work consists mainly from three phases as shown in *Fig 1*. Client connection goes through these phases to authenticate itself at the target service. These phases are as follows:

Phase 1: Secured communication initialization between client and IPS is accomplished. IPS validates the request and generates a client ID to be used through the next phases. Knocking sequence generation process is initiated at client side upon receiving that ID.

Phase 2: IPS informs securely the requested target server, which holds the target service, about the expected client connection. The target service initiates the same process as the client to generate the same knocking sequence.

Phase 3: Dynamic knocking session is initiated by the client to authenticate itself at target service. If successful knocking, client is allowed to use the target service, otherwise, client receives no response and IPS is informed. In addition, synchronization between client and service, due to lost and out of order of sent UDP knock packets, is considered in this phase.

First and second phases are invoked only at first client interaction with the target service, or when client reaches the end of a previously generated knocking sequence. Knocking sequence could be used for several future sessions between client and target service directly (phase 3) which decreases the overall system overhead.

In this work, IPS and target service are cooperated to protect the system from several types of attacks like DoS, brute force, and replay attacks. IPS internal intrusion model and information exchanged about malicious knocking behavior with the target service improves system protection against these attacks.

In the following sub sections, a detailed description of each communication phase is presented.

Phase 1: Communication initialization

In this phase, client and IPS are agreed on a shared key to secure the communication. IPS replies with client ID as well upon successful connection. This ID is concatenated with seed for generating knocking sequence for future usage. This process includes the following steps:

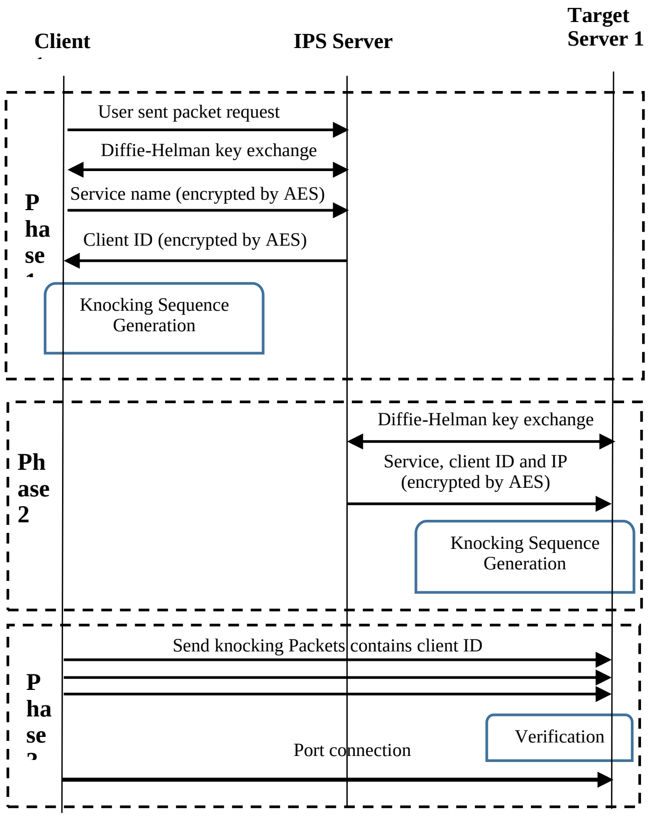
Client
IPS Server
Target Server 1
Phase 1
User sent packet request
Diffie-Helman key exchange
Service name (encrypted by AES)
Client ID (encrypted by AES)
Knocking Sequence Generation
Phase 2
Diffie-Helman key exchange
Service, client ID and IP (encrypted by AES)
Knocking Sequence Generation
Phase 3
Send knocking Packets contains client ID
Verification
Port connection

Fig 1. Proposed Knocking Phases.

Step 1: User sends a packet to IPS to request communication initialization.

Step 2: Client and IPS starts to agree on a shared secret key to secure the future communication using Diffie-Helman algorithm.

Step 3: The requested target service name is mapped to target server IP and service port. Service name, which is known only to legitimate users, is sent to IPS encrypted with AES algorithm using the early generated shared secret key.

Step 4: Upon receiving correct target service name, IPS generates a unique client ID and sends it back to the client. If a wrong target service is received, IPS rejects the connection.

Step 5: If previous steps are done successfully, client starts to generate a knocking sequence based on the received ID. The generation process will be stated later.

Phase 2: Notifying target service

In this phase, A secure connection is established between IPS and target server to inform the server about the expected connection. This includes the following steps:

Step 1: A secure connection is established using AES between IPS and target server with the help of Diffie-Helman algorithm.

Step 2: All information (Client IP and target service port) related to client request plus the generated client ID are passed to the target server.

Step 3: A sequence generation process is initiated at server side which is the same process as at client side. At this point, target service is ready to accept knocks from the requesting client.

Knocking sequence generation algorithm

In this section, knocking sequence generation process is described. The basic idea depends on using client ID and additional seed string as a base for sequence generation. This additional seed is known only to legitimate users and target service. Service name may be used as a seed for simplicity as it is already known to both legitimate users and target service. After generating the sequence, it is divided into knocking blocks which include a variable number of knocks with different knock ports for each block. The generation process steps are as follows:

Step 1: Hashing the concatenated client ID and the chosen seed using SHA-512. The output contains Ascii characters in range of [0 – 9] and [a-f].

Step 2: Substitute characters in range [a – f] with the corresponding values in range [10 – 15] to obtain the knocking sequence string of numeric Ascii characters only.

Step 3: Obtaining future connection sessions knocking blocks by dividing knocking sequence into several knocking blocks as follows:

Step 3-1: Let N_{min}, and N_{max} are the selected minimum and maximum number of knocks in each knocking block. Let R is actual number of knocks in knocking block. Let $P_{max} = 65535$ which is the maximum number of UDP ports. Let D represents one digit in the knocking sequence string and digit remainder $D_r = D \bmod N_{max}$.

Step 3-2: To specify next knocking block with R knocks, Traverse the generated string digits from left to right. Select R equals to D_r if ($D_r >= N_{min}$), else ignore and move to the next digit until a match exists.

Step 3-3: Select the next $R \times 5$ string digits to represent R knocking port numbers in the next knocking block considering each port number (P) consists of 5 digits string (P_s) and actual port number $P = P_s \bmod P_{max}$.

***Step 4*:** Repeat *step 3-2*, and *step 3-3* to get the next knocking blocks in the sequence.

The following example in (Table 1) clarifies how knocking sequence generation process works. Client ID is generated using UUID method as "a10050b8-38e0-478c-a1a4-932c70cdb32f". Used seed which should be target service name is chosen to be "this is the seed" in this example. N_{min} *and* N_{max} *is selected to be 2, 5 respectively*.

TABLE 1. EXAMPLE OF KNOCKING BLOCKS GENERATION PROCESSES

Concatenation	**a10050b8-38e0-478c-a1a4-932c70cdb32fthis is the seed**
Step 1: SHA-512	***952EC88A8ACC56ABD0792FA5C614C01CAD5E4AD953978B87B6D435B7460F922EF25681CE623A7878BE245F1552F048A40C1D284A018823E32EBCE9EA2C254783***
Step 2: Substitution of ***[a – f] by [10 – 15]***	***952141288108101212561011130792151051261412011210135144101395397811871161343511746015922141525681121462310787811142451515521504810401211328410018823143214111214914102122 54783***
Step 3- 2, 3-3: Knocking blocks. ***Blue Digits: Used to get number of knocks R = D mod*** N_{max}***.*** ***Red Digits: Ignored as each digit mod*** $N_{max} < N_{min}$	***9 52141 28810 81012 12561 (9 mod 5 = 4 knocks)*** ***0111 (Ignored as each digit mod*** $N_{max} < N_{min}$***)*** ***3 07921 51051 26141*** ***2 01121 01351*** ***4 41013 95397 81187 11613*** ***4 35117 46015 92214 15256*** ***8 11214 62310 78781*** ***11*** ***4 24515 15521 50481 04012*** ***11*** ***3 28410 01882 31432*** ***1*** ***4 11121 49141 02122 54783***
Step 3-4: Actual UDP knocking ports in each block by applying $P = P_s \bmod P_{max}$	***Knock block1 ports: 52141, 28810, 15477, 12561*** ***Knock block2 ports: 7921, 51051, 26141*** ***Knock block3 ports: 1121, 1351*** ***Knock block4 ports: 41013, 29862, 15652, 11613*** ***Knock block5 ports: 35117, 46015, 26679, 15256*** ***Knock block6 ports: 11214, 62310, 13246*** ***Knock block7 ports: 24515, 15521, 50481, 4012*** ***Knock block8 ports: 28410, 1882, 31432*** ***Knock block9 ports: 11121, 49141, 2122, 54783***

Phase 3: Knocking and authentication

In this phase, client tries to authenticate himself to the target service using the previously generated knocking sequence (Fig 2). A successful knocking attempt includes the following steps:

Step 1: Client selects the next knocking block in the generated sequence and starts to send UDP knocks to the target service. Each knock packet includes knock port as destination port, client ID, and packet order inside knocking block.

***Step 2*:** Upon receiving knocks at target server and based on the information received previously from the IPS about the expected connection, target server checks source address and client ID in each knock without giving any response to the client. If the received knocks match the current knocking block of this client at server side, the target service will be available to the client. If not, the connection is blocked by target server firewall and IPS will be informed.

***Step 3*:** Upon successful knocking, current knocking block is marked at both sides as the last used knocking block in the sequence.

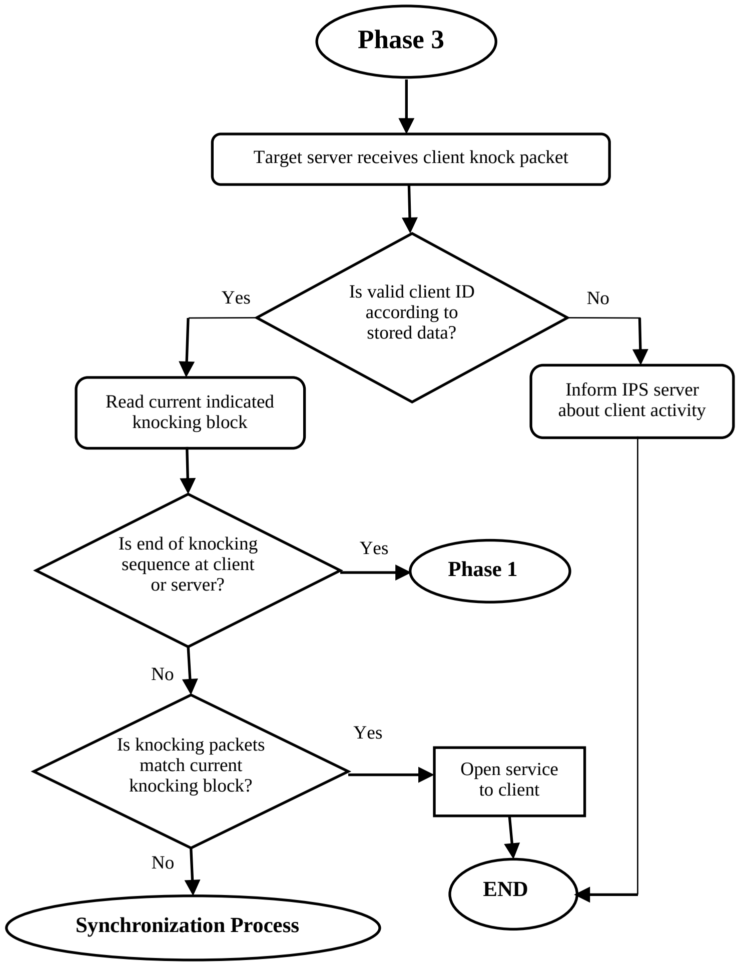
Phase 3
Target server receives client knock packet
Is valid client ID according to stored data?
Yes
No
Read current indicated knocking block
Inform IPS server about client activity
Is end of knocking sequence at client or server?
Yes
Phase 1
No
Is knocking packets match current knocking block?
Yes
Open service to client
No
END
Synchronization Process

Fig 2. Phase 3 knocking validation at target server.

Knocking challenges

Several issues are raised when applying the proposed knocking approach. Next chosen knocking block synchronization between client and target service, testing server connectivity, and knocking attacks are challenges that need to be considered. These challenges are handled as described below.

i. Knocking synchronization

Due to receiving out of order knocks and lost knocks which results in out of sync for current chosen knocking block at both client and target service, synchronization is a must to enable correct knocking behavior.

Out of order issue is handled by adding knock order number inside the sent knock. Receiver has the ability to reorder knocks based on this number. Sniffing the knocks order is not useful to attackers as the knocking blocks are changed dynamically.

Lost knocks are detected by the target service based on time out threshold. If loss is detected, server will ignore the current knocking block and move to the next one with no reply to the client (Fig 3).

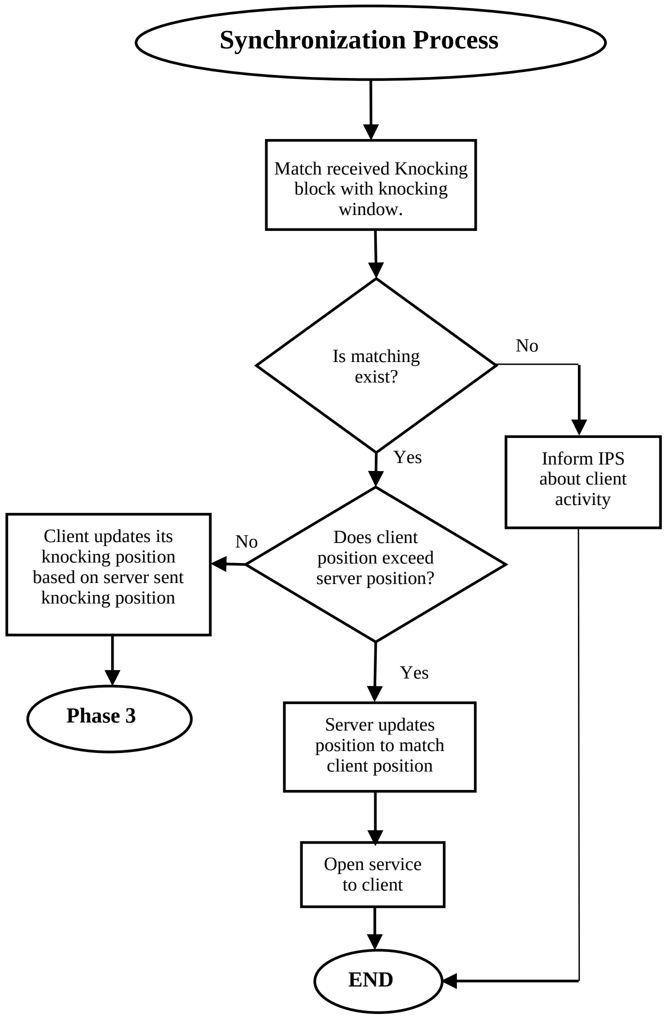
Synchronization Process
Match received Knocking block with knocking window.
Is matching exist?
No
Yes
Inform IPS about client activity
Client updates its knocking position based on server sent knocking position
No
Does client position exceed server position?
Yes
Phase 3
Server updates position to match client position
Open service to client
END

Fig 3. Client and server knocking synchronization process.

Lost knocks causes out of sync in current chosen knocking block between client and target service. This issue is handled at server side by using a matching window. If received knocking block is not matching the current knocking block, server tries to match it with a window of knocking blocks with the current knocking block in the middle of it. If no match, no response is provided and IPS is informed. If matching occurs at future part of the window, server will update its current position to be the same as client, otherwise client is informed to move its position to match server current position in the sequence. This process is shown in *Fig 3*.

ii. Testing target server connectivity

In port knocking, server presents no reply to incorrect knocking attempts. Client may need to test if the server is reachable without wasting knocking blocks in the sequence as each block is only used once and knock loss may occur. As mentioned earlier, in case of out of sync between client and server, successful matching with old part of server window results in server response with current knocking position in the sequence. Client may test connectivity by sending old knocking blocks within server window limits and capture response if exists. If no response, client tries server connectivity after a bit longer time. As reply only includes current server knocking position, there is no risk as knocking sequence is not known to attackers.

Knocking attacks

In this work, IPS and target service are cooperated to protect the system from several types of attacks. IPS internal intrusion model and information exchanged about malicious knocking behavior with the target service improves system protection against these attacks. Our proposed work can defense against the following attacks:

Replay attack: This type of attack repeats previously transmitted data between client and server to gain some benefits. Basic port knocking is exposed to replay attack by resending static knocking packets to server. The proposed approach depends on dynamic port knocking where number of knocks and knocking ports are changed with each connection session. So repeated knocking blocks will be ignored by the server.

Brute forcing attack: As each client generates its own knocking sequence based on the securely obtained client ID from IPS server, guessing the knocking sequence or correct knocking block is a challenging process to the attacker. IPS will block client immediately upon being informed by target service that an incorrect knocking block is received. This means that the attacker has only one chance to guess knocking. Knocking permutations and guessing probability are calculated as below.

Let

N_{min},N_{max}: Minimum and maximum number of packets in each knock block. Chosen here to be 2, 5 respectively.

n: Available port choices which are 65535 port number.

Available knocking permutations are computed as:

$$ {}_{n}P_{\;Nmax,\;Nmin} = \frac{n!}{(n-Nmax)!} - \frac{n!}{(n-Nmin-1)!} = \frac{65535!}{(65535-5)!} - \frac{65535!}{(65535-1)!} = 1{,}208{,}649{,}142{,}377{,}930{,}202{,}087{,}305 $$

Guessing probability is calculated as:

$$ Pb(n, Nmax, Nmin) = \frac{1}{n\boldsymbol{P}Nmax,Nmin} = 8.274\mathrm{e}^{-25} $$

Other types of attacks: As IPS server monitors traffic between client and target service, IPS can detect several types of attacks at different situations based on its internal detection model and received information from target server. Examples of these detection situations are as follows:

Connection initiation request: IPS server receives user connection request at the beginning of communication. If pervious malicious behavior of this user exists or request has invalid information like wrong service name, IPS server blocks the connection.

Incorrect knocking: If received knocking doesn't match any knocking block within server knocking window, server will inform IPS with user's information to block future user requests.

DoS attacks: Based on internal IPS model and collected statistics from target server about wrong knocking, flooding attacks, like DoS attack, can be detected and blocked. These statistics include how many wrong knocking trials is received within a time window. IPS use the received statistics to detect DoS attacks based on a preconfigured threshold. Hence, the cooperation between target server and IPS enable to handle such attacks.

IV. RESULTS AND ANALYSIS:

In this section, the performance of the proposed dynamic port knocking is evaluated using two different connection scenarios. This performance is evaluated in terms of server average processor utilization, server average memory utilization, and average knocking response time for each knocking block. Knocking block response time measures the interval between receiving first knock and sending successful knocking response at server side. Helper tools, like Top-like utility, PCP (Performance Co-pilot), and Tcpdump, are used for measuring performance metrics. These measurements are collected and analyzed using implemented bash script.

Experiments are executed on physical machine with Intel i7-5500U @ 2.40GHz processor and 8 GB memory. Three Linux virtual machines are built using VMWare 12 hypervisor on top of the physical machine. First virtual machine is the client machine (1 virtual processing core, 3 GB memory) which generates connection requests according to the applied scenario. Second machine (1 virtual processing core, 2 GB memory) plays the role of IPS server using Suricata engine and implemented python scripts. The third machine is the target server (4 virtual processing core, 2 GB memory) which receives and validates knocking blocks.

Scenario 1:

In this scenario, incremental number of users, from 10000 to 100000 users with step of 10000 users for each run, is conducted to generate a bulk of knocking requests to target server in minimum possible time interval for estimating workload impact on knocking server utilization. Users are generated using python code with multithreading. Each user is considered to send only one knocking block from its generated knocking sequence. Number of knocks in each knocking block is set to be within [1 to 4] range. Number of users at each run along with their corresponding total number of sent packets are stated in Table 2.

TABLE 2.FIRST SCENARIO: SENT PACKETS OVER EACH GROUP OF USERS.

Users	**10000**	**20000**	**30000**	**40000**	**50000**	**60000**	**70000**	**80000**	**90000**	**100000**
Packets	**29,337**	**57,950**	**88,110**	**117,152**	**143,576**	**173,077**	**202,530**	**231,571**	**261,392**	**290,045**
Phase 1,2 Avg Time per user (ms)	**11.3**	**11.3**	**11.6**	**11.1**	**11.5**	**11.1**	**11.4**	**11.5**	**11.5**	**11.6**

Overall average processing time per user is stated at the third row in Table 2. The processing values appears to be stable (about 11 ms) with different user groups which provides light initialization.

Fig (4) shows the relation between number of users in each run and total time taken by the server to process all users requests. It is obvious that time increases as users number increases due to more knocks are received and queued waiting for processing.

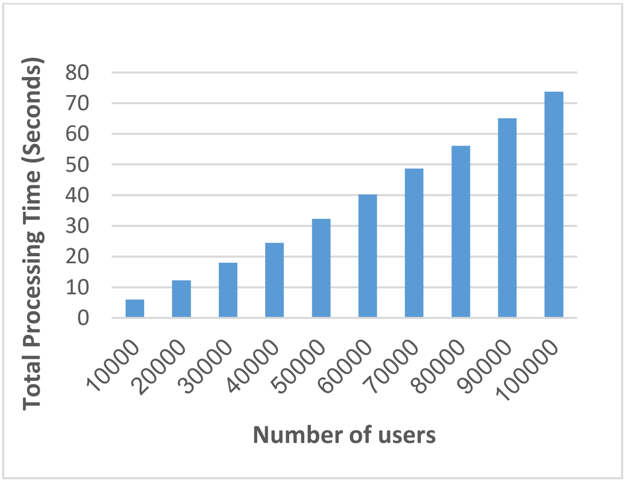

Fig 4. Relation between number of users and total knocking time in seconds at target server.

Server average processing rate of knocking blocks decreases as number of users increases at each run as shown in Fig 5. Average processor utilization shown in Fig 6 follows the same behavior as average processed knocking blocks. This is reasonable as increasing number of users results in increasing server received packets. Hence, an increasing part of processor time is consumed in handling and queueing network traffic which affects available time assigned to process knocking requests.

Memory utilization is increased gradually with increasing number of users as shown in Fig 7. Hashing map is used to store user's information which improves searching process. User information includes client ID, knocking sequence, current knocking position, and user address information. Memory utilization increases linearly with only 0.1% (0.001 x 2GB of total server memory = 2 MB) for each newly added 10000 users at each step. This shows minimal memory footprint is needed to deal with more users.

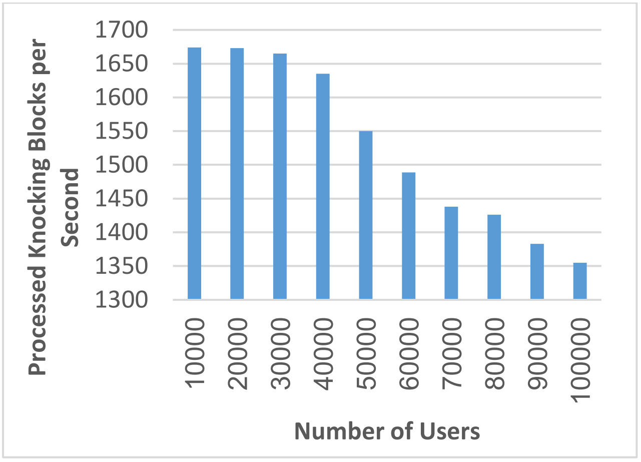

Fig 5. Relation between number of users and average processed knocking blocks per seconds.

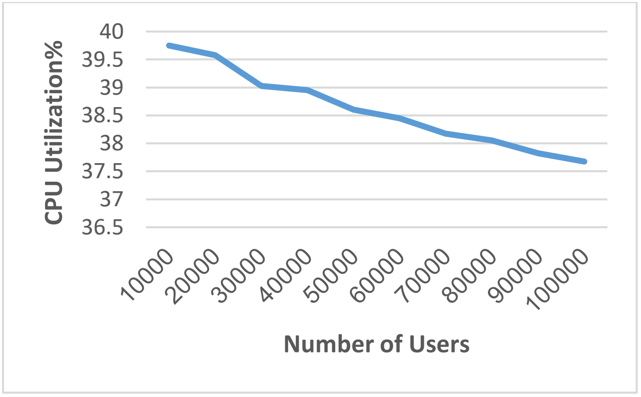

Fig 6. Relation between number of users and average processor utilization at server side.

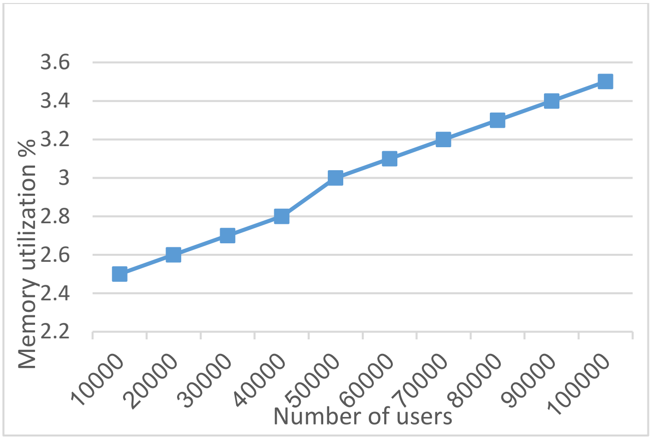

Fig 7. Relation between number of users in the system and average memory utilization at server side.

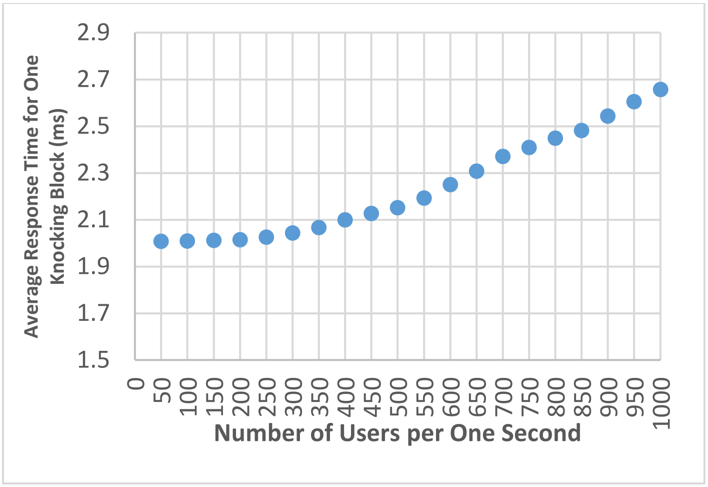

Fig 8. Average knocking block response time at server side against number of users per second.

Scenario 2:

In this scenario, system is fed with specific number of users in one second. Users are generated using python code with multithreading. Users' number starts from 50 to 1000 users per second with incremental step of 50 users for each run. Other configurations are set as mentioned in scenario 1. Table 3 presents the number of users per seconds at each run and their corresponding total number of sent packets.

In Fig 8, Average CPU utilization of the server increases linearly as number of users per second added to the system increases. As number of processed packets is not big enough to push the server to its processing limits, more processor time can be assigned to process knocking blocks while handling and queuing users' traffic. Average knocking block response time increases gradually as well as shown in Fig 9. This is reasonable due to more requests are queued until picking it up to be processed. Even at maximum load of 1000 users/second, server average response time is still in the range of several milliseconds (about 2.7 milliseconds at 1000 users/seconds

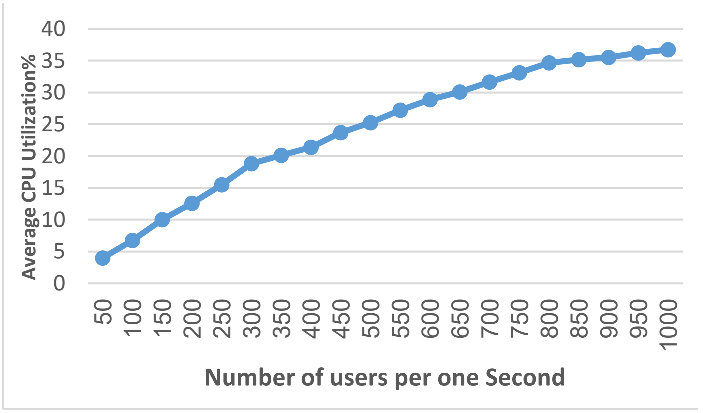

Fig 9. Average CPU utilization at server versus number of submitted users per second.

Comparison between several research efforts and the proposed work is stated in (Table 4) based on provided features. The proposed approach provides most of features compared to other approaches. The proposed knocking is dynamic with multiple knocks in each authentication trial which hardens the proposed approach. It presents authentication per service and not for the whole server running services. Protection against several attacks is provided through the cooperation between IPS server and target server. Packet loss and out of order are handled. Testing service reachability is considered as well.

Processing overhead comparison is based on three operation complexity levels which are high, medium, and low. High level is assigned to approaches that include image processing and certificate authentication. Medium level is assigned to those which use cryptography based approaches. At last, low level is assigned to approaches that use clear information in their knocking. Processing overhead of the proposed approach is low compared to others. This is because first and second phases are invoked only at first client connection. Once knocking sequence is generated, it is used for multiple future sessions. In addition, clear knocking information is used without the need to complex operations like image processing and extensive use of cryptography.

TABLE 3: SECOND SCENARIO: SENT COLLECTION OF USER'S PACKETS OVER ONE SECOND.

Users/s	**50**	**100**	**150**	**200**	**250**	**300**	**350**	**400**	**450**	**500**	**550**	**600**	**650**	**700**	**750**	**800**	**850**	**900**	**950**	**1000**
Packets	**151**	**295**	**441**	**585**	**733**	**886**	**1033**	**1181**	**1323**	**1468**	**1617**	**1760**	**1913**	**2057**	**2261**	**2434**	**2549**	**2692**	**2835**	**2964**

TABLE 4. COMPARISON BETWEEN THE PROPOSED WORK AND OTHER RESEARCH EFFORTS.

Comparisons	**Need for external servers**	**Authentication per server or service?**	**Brute force immunity**	**Replay attack immunity**	**DOS Hardening**	**Testing Connectivity**	**Out of order solving**	**Packet loss handling**	**Processing overhead**	**Dynamic knocking**	**Multiple knocking packets**
Basic port knocking [1]	No	Server	Yes	No	No	No	No	No	Low	No	Yes
One time knocking using SPA and IPsec [6]	Yes	Server	Yes	Yes	No	No	No	No	Medium	Yes	No
Hybrid port knocking [7]	Yes	Service	Yes	Yes	Yes	No	Yes	No	High	No	Yes
Port knocking with QRC and AES [8]	Yes	Server	Yes	Yes	Yes	No	No	No	Medium	Yes	Yes
Port knocking against TCP replay [9]	No	Server	Yes	No	No	No	No	No	Low	No	Yes
Secure port knocking tunneling [10]	No	Server	Yes	No	Yes	No	No	No	Low	No	Yes
sKnock: Port knocking for masses [5]	Yes	Service	Yes	Yes	No	No	No	Yes	Medium	No	No
Hiding from automated network scans [3]	Yes	Server	Yes	Yes	No	No	No	No	Medium	Yes	No
Authentication for access mobile web service [4]	Yes	Server	Yes	Yes	Yes	No	Yes	No	High	No	Yes
Building disguised server using honey port [2]	Yes	Service	Yes	Yes	No	No	No	No	High	Yes	No
Proposed approach	**Yes**	**Service**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**Low**	**Yes**	**Yes**

V. CONCLUSION AND FUTURE WORK:

In this research, dynamic knocking approach is proposed. It consists of three phases. At first phase, user requests connection to a specific service by providing correct service information securely to Intermediate IPS. Then client receives IPS response to able to generate a unique knocking sequence. At second phase, IPS informs securely the target service with user future client connection intention and provide client information to it. Then, target service generates the same unique knocking sequence as the client. At third phase, client knocks the service with correct knocking block to gain access to it. The first and second phases are used only once at the first connection. Knocking sequence is used for several future knocking sessions. Knocking block has variable number of knocks and different knock ports in each knocking session. This results in robust and low overhead knocking approach.

The performance of the proposed work is evaluated using two scenarios. The first scenario generates knocking requests using a bulk of users up to 100000 users while the second scenario generates knocking requests by feeding the system with users of rate up to 1000 users/second. The results show that the proposed approach has minimal memory footprint, and average server knocking response time of p2.7 milliseconds at full load.

The proposed approach provides several features compared to other approaches. Cooperation between IPS and target service is implemented to provide protection against malicious activities like DoS attack. Protection against replay and brute force attacks are provided as knocks number and ports are always changed. Current knocking position synchronization between client and server is considered using matching window to overcome issues like packet loss and out of order knocking. Testing service reachability is another feature that enables client to check connectivity without wasting sequence knocking blocks.

As a future work, extending the scalability of the dynamic knocking authentication service should be studied. The service could be implemented at a dedicated server to handle port knocking authentication at all phases. Furthermore, distributing the service at several servers would enable the service to be more scalable and serve more requests in highly loaded domains.

REFERENCES

[1] M. Krztwinski, "Port knocking Network Authentication Across Closed ports," *SysAdmin Magazine,* p. 4, June-2003.

[2] H. S. Park, Y. B. Jeon and Y. J. Won, "A new approach to building a diguised server using the honey port against general scanning attacks," *Springer International publishing,* p. 13, 2017.

[3] K. William and B. Azer, "PROVIDE: Hiding from Automated Network Scans with Proofs of Identity," in *2016 Fourth IEEE Workshop on Hot Topics in Web Systems and Technologies,* Boston, 2016.

[4] K. Mohamed, D. Wijesekera and P. C. Costa, "An Authentication Mechanism for Accessing Mobile Web Services," *Springer International Publishing,* p. 15, 2017.

[5] D. Sel, S. H. Totakura and C. Georg, "sKnock: Port knocking for Masses," in *IEEE 35th Symposium on Reliable Distributed Systems Workshops*, 2016.

[6] J.-H. Liew, S. Lee, I. Ong, H.-J. Lee and H. Lim, "One-Time Knocking framework using SPA and IPsec," in *2010 2nd International Conference on Education Technology and Computer*, Shanghai, China, 2010.

[7] H. Al-Bahadili and H. H. Ali, "Network Security Using Hybrid Port Knocking," *IJCSNS International Journal of Computer Science and Network Security,* p. 4, 2010.

[8] V. Srivastava, A. K. Keshri and A. D. Roy, "Advanced port knocking authentication scheme with QRC using AES," in *2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)* , Udaipur, India, 2011.

[9] F. H. M. Ali, R. Yunos and M. A. M. Alias, "Simple port knocking method: Against TCP replay attack and port scanning," in *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on* , Kuala Lumpur , 2012.

[10] P. Mehran, E. A. Reza and B. Laleh, "SPKT: Secure Port Knock-Tunneling, an enhanced port security authentication mechanism," in *2012 IEEE Symposium on Computers & Informatics (ISCI),* Penang, Malaysia, 2012.

[11] B. Mahbooba and M. Schukat, "Digital Certificate-based Port knocking for connected Embedded Systems," in *2017 28th Irish Signals and Systems Conference (ISSC),* 2017.

[12] M. Khader, A. Hadi and A. Hudaib, "Covert Communication Using ort Knocking," in *2016 Cybersecurity and Cyberforensics Conference*, Amman, 2016.

[13] F. v. Eye, M. Grabatin and H. Wolfgang, "Detecting Stealthy Backdoors and Port Knocking Sequences through Flow Analysis," *DE GRUYTER,* p. 97–104, 2015.

[14] H. Liu, Z. Wang and Y. Liu, "Address Knocking: An Undetectable Authentication Based on IPv6 Address," in *2012 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Beijing, China, Dec. 2012.

[15] R. deGraaf, J. Aycock and M. Jacobson, "Improved port knocking with strong authentication," in *21st Annual Computer Security Applications Conference (ACSAC'05)*, Tucson, AZ, USA, 2005.

[16] F. Yunos, M. Hani and R. Ali, "Simple Port knocking Method," in *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on*, Kuala Lumpur, 2012.

[17] K. Z.A., J. N, A. M.H. and B. A., "Performance Evaluation of widely used Portknocking Algorithms," in *High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on*, Liverpool, 2012.