Associating the Authentication and Connection-Establishment Phases in Passive Authorization Techniques

Muhammad Tariq, M. S. Baig, M. Tariq Saeed

***Abstract*—Port Knocking and Single Packet Authorization (SPA) are passive authorization techniques that provide firewall-level authentication to ensure authorized access to potentially vulnerable network services. Although these techniques serve as powerful protection mechanism against the intruders, they still suffer from a major problem. The lack of association between the authentication process and the follow-on TCP connection to be established is the most crucial problem still persisting in both of the passive authorization techniques. This problem allows an attacker to connect to a protected server on behalf of a valid client, after the client has successfully authenticated with the firewall but before he establishes a TCP connection with the protected server. In this paper, we propose a novel design for the resolution of this problem. We show that the design is incorporable into the existing architectures of both passive authorization techniques while keeping the previously made enhancements to these systems intact. For the purpose of performance evaluation, we have extended the capabilities of an existing open-source port knocking system, JPortKnock. Simulation results show that the processing overhead, which is crucial for passive authorization systems, incurred by incorporating this design into the existing systems is marginal.**

***Index Terms*—Dynamic Firewalls, Passive Authentication, Port Knocking, Single Packet Authorization.**

I. Introduction

The two passive authorization techniques of Port knocking and Single Packet Authorization (SPA) provide firewall-level authentication for remote users and the capability of dynamic reconfiguration of firewalls. Port knocking [1], [2] is a method of opening closed ports on a firewall, configured in all-drop mode, by sending a series of TCP or UDP packets on pre-specified ports. The crude idea was vulnerable to many problems [3], [4], [5] but it has matured over the time with cryptographic [6] and steganographic enhancements [7] to the concept and various other improvements [3], [8]. Later MadHat [9] introduced 'single packet authorization'. The functioning of SPA [10], [11] is the same as port knocking except that instead of using multiple TCP and UDP packets, the authentication data is put in the payload of a single UDP packet. Both techniques have their own pros and cons [12] and they exist side by side. More than 30 implementations are available for both port knocking and SPA [13].

Though port knocking and SPA have matured over the time, the authentication-connection association problem [3], [12] which is the most crucial issue concerning both of these techniques still persists. The literature so far surveyed shows no significant attention paid to this problem. It is easy for an attacker, given adequate position, to exploit this vulnerability. The result of this exploitation is total bypassing of the authentication at firewall making the entire concept useless.

The authentication-connection association problem is basically a disjoint between the authentication phase and the subsequent TCP connection establishment phase in both passive authorization techniques. It is possible for an attacker, in a position to observe connections between client and server and the typical information associated with those connections, to wait for the authentication process to complete first and after that take the client down, either by launching a denial-of-service attack or by any other means, and connect to the server impersonating as that client as shown in fig. 1. The problem becomes more critical if the attacker lies in the private network of the client behind a NAT, sharing a single public IP address with that client. In such scenario, after waiting for the authentication process to complete, the attacker just needs to send the TCP connection establishment request to the server before the client. The attacker does not even need to take the client down in such a scenario and can connect to the server on client's behalf.

Session-hijacking [14], [15] is a method of attack where an attacker takes control over an established TCP session to execute his commands on the server on behalf of the authenticated client in the session. In this paper we are only concerned with resolving the authentication-connection association problem. Session hijacking is a separate problem and is out of the scope of this paper.

The rest of the paper is organized as follows: Section 2 presents the prior work on resolution of this problem. Section 3 presents the proposed design for resolving the authentication-connection association problem along with

Manuscript received March 14, 2008. The author is very pleased to give warm thanks to National University of Sciences and Technology for the valuable support in carrying out this research work and for the financial assistance.

Muhammad Tariq is with the Center for Cyber Technology & Spectrum Management, National University of Sciences and Technology (NUST), Pakistan (phone: +923345055993; e-mail: mtariqpk@hotmail.com).

M. S. Baig is with the Center for Cyber Technology & Spectrum Management, NUST, Pakistan (e-mail: msbaig_nust@hotmail.com).

Muhammad Tariq Saeed is with Military College of Signals, NUST, Pakistan (e-mail: tariq.programmer@gmail.com).

the underlying assumptions. Section 4 discusses the performance evaluation of our proposed system. Section 5 concludes the paper and gives directions for future work.

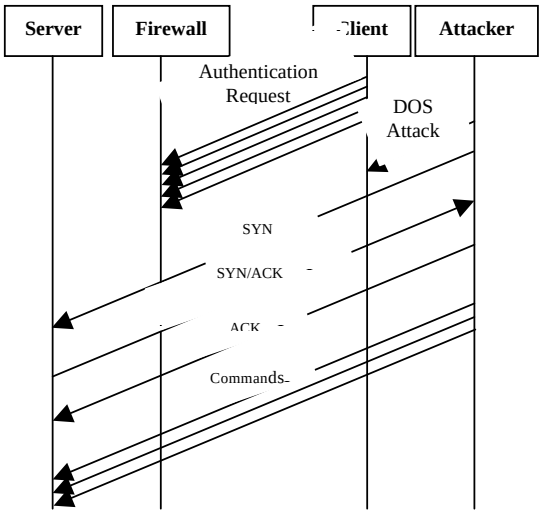

Fig. 1: Series of steps an attacker follows to exploit the authentication-connection association problem in port knocking

II. Prior Work

Not a lot of work is available in the literature to resolve the authentication-connection association problem. This problem is also referred to as 'race attacks' in the literature. Except Tan and Cappella's system [16], all other solutions proposed to resolve this problem are in the form of suggestions to be implemented as future extensions for the work of various authors. Most of the previously proposed solutions either compromise the basic objectives of passive authorization systems i.e. simplicity and lightweight, or they are unfeasible to be implemented in real time systems. Tan and Cappella proposed a partial solution to the authentication-connection association problem using a 5-step procedure. In their system, the server informs the client in an encrypted UDP packet about the random port at which the service would be available for that client and then expects the connection attempt on that random port. Their technique does not prevent the attackers from locating the random port by scanning or discovering it by blocking the client after successful authentication and sending the TCP connection request on every port. Their technique is also susceptible to the loss of stealth, which is a basic necessity of passive authorization techniques, because the server generates a response to send the port number to the client and that may allow attackers to notice the existence of the server.

DeGraaf [3] proposed to introduce a secondary wrapper server within the architecture which, on successful authentication of client, would tunnel the post-authentication connection to the actual server. This idea adds too much complexity at the server-end compromising the basic requirements of simplicity and lightweight. DeGraaf also suggested negotiating the ISN (Initial Sequence Number) to be used in the subsequent connection during the authentication phase. This idea is unfeasible to be implemented in real time systems as it would require that client and server are implemented in operating system's kernel space.

Jeanquier [12] proposed to wrap the post-authentication connections within encrypted sessions but the added cryptographic operations required for doing that will overload the server, compromising the basic objectives of passive authorization schemes.

All of the ideas discussed above are either computationally intensive, require the clients to have unrealistic or unfeasible privileges from operating system kernels or they add excess of complexity on top of the initially simple concepts of port knocking and SPA.

III. Proposed Design

Before we proceed to our proposed design, we list down the assumptions that underlay our solution.

A. Assumptions

1. We assume that the attacker possesses the power to craft and send any type of packets containing any information and these are not blocked by ingress or egress filters.
2. We assume that it is hard for an attacker to sniff the ISN (Initial Sequence Number) and ACK (Acknowledgement Number) from packets in transit between client and server. However he can observe on-the-fly the connections between client and server as soon as they are established and the typical information associated with these connections like source and destination IP addresses of packets, source and destination port numbers, underlying network protocol (TCP, UDP, ICMP) and application-level protocols being used (e.g. SMTP, FTP, etc).
3. The attacker cannot stop or modify the packets in transit between the client and server.

It is important to note that it is trivial for an attacker to exploit the authentication-connection association problem even if he does not have the access to complete information contained in the packets being transmitted between client and server. The authority of observing connections made between client and server on-the-fly and the typical information associated with those connections is sufficient for an attacker to exploit this vulnerability. An example of an attacker in such a position could be of one having user-level access to a router, firewall or a proxy server near the client or having access to the log files generated by routers and firewalls on-the-fly. For an attacker who is able to sniff all the information from the packets in transit between client and server, it is not difficult to even hijack an established TCP session, not to mention the authentication-connection association problem. This weakness is a legacy of TCP because security was not an issue when TCP was developed. In this paper, we are not concerned with the attackers possessing this power.

B. Description of Proposed Approach

A major challenge in proposing a solution to the authentication-connection association problem is to devise a design that would not compromise the simplicity of passive authorization schemes and keep these architectures lightweight. We have come up with such a solution by sharing something as simple as a nonce between client and server. The idea is to exchange an encrypted nonce between the client and the firewall during the authentication phase and then encode that nonce in a suitable IP or TCP option field in the SYN packet of the subsequent TCP connection establishment phase, to associate the two phases. After a successful authentication, the firewall will only accept that connection attempt which would be carrying that nonce in its first request packet. The connection establishment request packets of the attackers without the nonce, shared during the authentication phase, would be dropped by the firewall.

We propose two implementation alternatives for our design. The first approach, in section C, presents an effective way to incorporate our design in those port knocking systems that rely on challenge-response [3], [16] during the authentication phase. Section D presents the second approach for the typical implementations of port knocking and SPA systems that do not rely on challenge-response during the authentication process.

C. For Challenge-Response based Port Knocking Systems

As already stated, the notion of challenge-response already exists in a few port knocking systems. The first approach incorporates our design in such systems, as shown in algorithm 1. **A** sends the authentication request to **F**, using multiple packets in case of port knocking and using a single packet in case of SPA. **F** checks the validity of the request, if the request is not valid no response is sent and the packet is dropped. In response to a valid request, a 64-bit nonce is randomly generated, encrypted using a shared symmetric key and is sent to **A** in the payload of the UDP packet used by challenge-response based port knocking systems to send information to the client. This response serves as an acknowledgement confirming **A** that the pre-specified port on **F** has been opened. **A** decrypts the nonce, encodes it in an appropriate IP or TCP options field in the SYN packet of the subsequent TCP connection and sends it to **B** through the open port. The SYN packet is allowed through **F** to **B** as it carries that nonce. Every connection request after successful authentication at **F** would be dropped if it does not contain the nonce shared between **F** and **B**. **B** sends back the SYN/ACK request normally, **A** Sends an ACK as a response and the TCP session is established successfully.

This approach is well suited to be incorporated into the challenge-response based architectures of port knocking [3], [16]. The security of our proposed design is dependent on the security of encryption algorithm used to encrypt the nonce sent as a challenge to the client. We propose to use AES for encryption with 128-bit key length.

It is important to mention that the client does not encrypt the nonce that he sends encoded in the SYN packet of the actual TCP connection. Even if the attacker is able to sniff that unencrypted nonce somehow, it would not work for him as the client's packet would have reached the server to initiate the session before the attacker crafts a packet and attaches that nonce to it, because we assumed that the attacker is not able to stop or modify a packet in transit between client and server. The attacker can block all the traffic from client though, by launching a DOS attack but in that case he will have to rely upon that encrypted nonce sent by the server and decrypt it to gain any benefit. The attacker may also take the client down after the client has sent the SYN packet, sniff that unencrypted nonce of that packet, send a spoofed RST packet to reset the client's session and initiate his own connection along with a valid nonce. To thwart such attempts, the firewall should allow only one connection attempt per successful authentication.

1: A → F: Auth
2: F → A: E_K (N_A)
3: A → B: TCP SYN, N_A
4: B → A: SYN/ACK
5: A → B: ACK

Where
- **A** is the client
- **F** is the firewall
- **B** is the server
- **Auth** is the authentication request
- E_K is the encryption key shared between A & F
- N_A is the nonce sent to A by F
- **TCP SYN** is the first packet belonging to the connection establishment phase
- **,** denote concatenation

Algorithm 1: Proposed Communication Protocol for Challenge-Response based Port Knocking Systems

D. For Standard Port Knocking and SPA Systems

The problem with first approach is that it requires the firewall to generate a response to the authentication request of a client and the majority of Port Knocking and SPA implementations do not follow this scheme. It is also not recommended in systems like port knocking and SPA as it may result in loss of stealth confirming to attackers about the existence of a listening server.

To incorporate our design with those port knocking and SPA schemes that do not rely on challenge-response during the authentication with the firewall, we propose a modification in the first approach, shown in algorithm 2.

1: A → F: Auth, E_K (N_F)
2: A → B: TCP SYN, N_F
3: B → A: SYN/ACK
4: A → B: ACK

Where
- N_F is the 32 bit nonce sent to firewall

Algorithm 2: Proposed Communication Protocol for Standard Port Knocking and SPA systems

In the second approach **A** instead of **F** is required to generate a random nonce, encrypt it using the shared symmetric key and send it to **F** along with the authentication request. In SPA this nonce can simply be made a part of encrypted payload. But in port knocking, because of increased size due to encryption, the encrypted nonce may be

divided in smaller parts and sent by encoding in suitable fields of the multiple request packets used by port knocking for authentication, as shown in fig. 2. These parts will have to be collected and merged by **F.** After validating the request and decrypting the nonce, **F** will open the pre-specified port for **A** to access the service. Like in most of the port knocking and SPA systems, the server generates no response thus preventing from the loss of stealth. **A** will then send unencrypted nonce encoded in the SYN packet of the subsequent TCP connection and the rest will go smoothly as already explained in first approach.

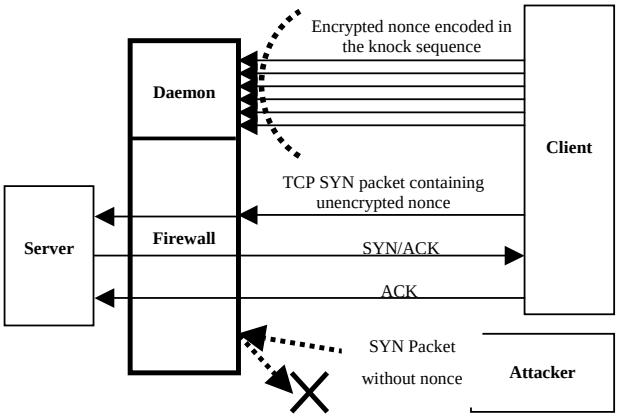

Fig. 2: Incorporating the proposed design in port knocking by encoding the encrypted nonce in suitable fields of knock sequence

In both port knocking and SPA, the clients use a program that acts as a wrapper over the service-client. This wrapper authenticates the client with firewall and keeps the entire procedure transparent from the service-client. For incorporating any of the two designs we have proposed, the wrapper would require some extension to its capabilities in order to act as a proxy for the service-client. All the traffic of the service-client would be tunneled through that proxy to allow the wrapper to encode nonce in the SYN packet of the connection to be established. The wrapper would also recalculate and update the checksum of the SYN packet. It is important to mention that this extension in the capabilities of the wrapper does not require any support from the operating system kernel and adds no complexity on the server-side, unlike most of previously proposed solutions.

For encoding the nonce in the SYN packet of the TCP connection to be established, IP Timestamp and TCP echo option are two suitable fields as proposed in [17]. Size of both fields is 36 bytes each that can easily accommodate an 8-byte nonce. Note that if the timestamp field is used for transmission of nonce, the timestamp buffer has to be flagged as full, so that intermediate routers do not manipulate the timestamp. To ensure end-to-end reliability in transmitting the packet, we recommend not using the payload of the SYN packet to transmit the nonce because some routers trim the payload attached to the SYN packets and intrusion detection and prevention systems filter SYN packets with large payloads. The size of nonce to be used should be atleast 32-bits but we recommend using 64-bit randomly generated nonce to ensure good security.

IV. Performance Evaluation

We evaluate the performance of our design on the basis of three parameters: robustness, processing overhead and stealthiness.

A. Robustness

Under robustness we analyze the effects of various intermediate devices, like firewalls, intrusion detection and prevention systems, proxies and NATs, on our proposed design. Some firewalls potentially trim the data attached to the payload of a SYN packet passing through it. Similarly, many IDSs and IPSs deem the SYN packets with payload as malicious. To deal with these issues, we have proposed to encode unencrypted nonce in suitable IP or TCP options field, during the actual TCP connection establishment phase, in both of the implementation alternatives of our design. Using the IP Timestamp field to encode data with the Timestamp buffer flagged as full, these blockades are evaded.

The ability of working across NATs/proxies has been a significant issue for both port knocking and single packet authorization. Some systems have partially got rid of this issue by adding various innovations to the concepts of passive authorization techniques but some of them still suffer. In case of our first approach, if the attacker and the client are behind a common NAT, the attacker can take the client down after the client has sent the authentication request to the firewall and receive the challenge packet containing encrypted nonce on behalf of that client. But since the nonce is encrypted, it would be of no use for the attacker because to connect to the server on behalf of that client, the attacker needs to decrypt that nonce. The second design alternative we have proposed is totally immune to this issue because in that the nonce generation, encryption and its transmission along with the authentication request is to be done by the client. Under our initial assumptions, it is impossible for an attacker to sniff the encrypted or unencrypted nonce from the packets sent by client to the firewall and server, thus preventing against the exploitation of the authentication-connection association problem.

B. Processing Overhead

Most of the problems that have been resolved in port knocking and single packet authorization based dynamic firewalls are at the cost of increased overhead on these systems. The more features are added to these systems, the more they become vulnerable to DOS attacks. Hence a primary concern while enhancing these systems is that the overhead imposed by the proposed enhancements at the server side remain marginal.

We have implemented our proposed design by modifying an existing open-source port knocking system, JPortKnock [18]. The test platform used for the evaluation of the simulation was a dual Intel Xeon machine running at 2.8 GHz with a 4 MHz bus, 512 kB L2 Cache and 1 GB of RAM.

To measure the performance overhead incurred by incorporating our proposed design in the existing systems, the ability of processing different numbers of simultaneous authentication request packets of an existing and the proposed system was tested and the deviation in corresponding delay was observed. Existing system used was the standard JPortKnock System and proposed system was

the version of JPortKnock in which our proposed design was incorporated. The results obtained after the experiments conducted during the implementation phase are shown in Table I & II. The percentage processing overhead imposed by the existing and proposed systems was calculated using the following formulas.

Table I: showing results obtained from the simulation

Auth. Requests Per Second	Delay in milliseconds		
	Theoretical Results (X)	Existing System (Y)	Proposed System (Z)
1	0.344	0.344	0.347
2	0.688	0.691	0.698
4	1.376	1.383	1.396
8	2.752	2.763	2.788
16	5.504	5.523	5.565

Table II: showing the percentage processing overhead of the proposed system

Percentage Processing Overhead of Proposed System	
Compared to Theoretical Results (DT)	Compared to Existing System (DE)
0.872 %	0.872 %
1.453 %	1.013 %
1.453 %	0.940 %
1.308 %	0.905 %
1.108 %	0.760 %

$$DT = \{(Z-X)/X\}*100$$
$$DE = \{(Z-Y)/Y\}*100$$

The percentage processing overhead imposed by our proposed design, compared to the theoretical results and the results obtained from the existing system, is shown in Fig. 3. The graph shows that the processing overhead of the proposed system when compared to the existing system remains less than 1% on average. The processing overhead of the proposed design when compared to the theoretical results also does not exceeds 1.5%.

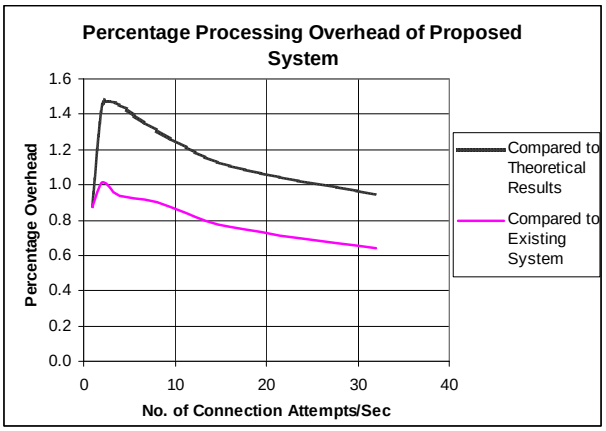

Fig. 3: Graph showing the percentage processing overhead of the proposed system as compared to theoretical results and results of existing system

The results clearly show that the overhead imposed by our proposed design over the existing system is very marginal as compared to the criticality of the problem of lack of association between the authentication process and the follow-on TCP connection to be established, in both port knocking and single packet authorization.

C. *Stealthiness*

Stealthiness is a characteristic that is defined as the ease with which an attacker can discover a hidden service. Primarily in both passive authorization techniques, the server sends no response until a valid passphrase/knock sequence is received. But some port knocking architectures that rely on challenge-response during the authentication phase are susceptible to the loss of stealth. Our first approach is proposed to be incorporated into such systems; it is as susceptible to the loss of stealth as those challenge-response based port knocking systems are. But even in these systems, our scheme uses encrypted nonce to be sent by the firewall to the client, it is still impossible for an attacker to get any benefit. However, our second approach has to be incorporated into completely non-responsive implementations of port knocking and SPA, this approach does not compromise the stealthiness of these systems. Use of encryption in both of the proposed implementation alternatives helps in ensuring stealth, against those attackers having limited capability of sniffing the information in transit between the client and firewall.

V. Conclusion And Future Extensions

In this paper, we presented how the authentication-connection association problem can be exploited by an attacker. We proposed a simple and lightweight design enhancement to the existing architectures that would make it impossible for an attacker to exploit this vulnerability and connect to the service on behalf of a valid client. We explained two approaches by which our solution can be incorporated into the existing architectures of port knocking and SPA without compromising their existing strengths. At the end, we showed that the processing overhead imposed by incorporating our design in existing systems is marginal. There are two problems that still exist in the passive authorization techniques, one is the susceptibility to DOS attacks and second one is the problem of working across NAT. These two problems offer the scope for future work in this domain. Extension of the powers offered by port knocking and single packet authorization schemes to hardware based dynamic firewalls also has the potential for future work.

References

[1] P. Lunsford, E. C. Wright, "Closed Port Authentication with Port Knocking", *Proceedings of the American Society for Engineering Education Annual Conference & Exposition*, 2005, pp 1747-1754.

[2] M. Krzywinski, "Port knocking: Network authentication across closed ports" in *SysAdmin Magazine*, 2003, pp 12–17.

[3] R. deGraaf, J. Aycock, M. Jacobson, "Improved Port Knocking with Strong Authentication" *IEEE/ACM Proceedings of the 21st Annual Computer Security Applications Conference*, 2005, pp 409-418.

[4] A. Manzanares, J. Marquez, J. Tapiador, J. Castro, "Attacks on Port Knocking Authentication Mechanism", *ICCSA LNCS-Springer Berlin/Heidelberg*, 2005, pp 1292-1300.
[5] A. Narayanan, "A critique of port knocking" in *NewsForge*, August 2004.
[6] David Worth, "COK - Cryptographic One-Time Knocking", in Black Hat Slides USA, 2004.
[7] E. Vasserman, N. Hopper, J. Laxson, J. Tyra, "SILENTKNOCK: Practical, Provably Undetectable Authentication", *ESORICS LNCS-Springer Berlin/Heidelberg*, September 2007, pp 122-138.
[8] Michael Rash, "Combining port knocking and passive OS fingerprinting with fwknop", in *USENIX ;login: Magazine*, December 2004.
[9] Madhat Unspecific, Simple Nomad, "SPA: Single Packet Authorization", 2005.
[10] Michael Rash, "Single Packet Authorization with fwknop" in *USENIX ;login: Magazine*, February 2006.
[11] Michael Rash, "Protecting SSH Servers with Single Packet Authorization" in *The Linux Journal*, May 2007.
[12] S. Jeanquier, "An Analysis of Port Knocking and Single Packet Authorization", MSc Thesis, University of London, September 2006.
[13] "Port Knocking". Available: www.portknocking.org
[14] Shray Kapoor, "Session Hijacking: Exploiting TCP, UDP and HTTP Sessions", July 2006. Available: www.infosecwriters.com/text_resources/pdf/SKapoor_ SesionHijacking.pdf
[15] "Theft on the Web: Prevent Session Hijacking", Available: http://www.microsoft.com/technet/technetmag/issues/2005/01/ SessionHijacking/default.aspx
[16] C. K. TAN, Cappella, "Remote Server Management using Dynamic Port Knocking and Forwarding" in *Special Interest Group in Security and Information Integrity*, May 2004.
[17] P. Barham, S. Hand, R. Isaacs, P. Jardetzky, R. Mortier, and T. Roscoe, "Techniques for Lightweight Concealment and Authentication in IP Networks," *Intel Research, Tech. Rep. IRB-TR-02-009*, July 2002.
[18] "JPortKnock". Available: https://jportknock.dev.java.net/