Port Knocking- An Additional Layer of Security for SSH and HTTPS

Jigar A. Raval[1], **and Samuel Johnson**[1]
[1]Computer Center, Physical Research Laboratory, Ahmedabad, Gujarat, India

Abstract - *The availability of communication resources remote access of e-mails and data is increasingly required and desirable by users. This, however, implies security of user data and e-mails. The vulnerability of the system depends on the ability to scan the system for an open port and on the service running on the open port. Such open ports are entry points for attacks. Masking of open ports and services on the system, using port knocking technique, provides a simple and reliable method. Initially, during the port knock sequence all the ports remains closed, thus the services or open ports become invisible to any malicious port scan. After a valid port knock, a predetermined port is opened allowing access of predefined service. Thus port knocking technique adds an extra layer of security without any major changes to the application.*

In this paper, we discuss a complete practical approach of securing SSH and HTTPS (Web based email access) using available open source software. We wish also to share our experiences so that enterprise level secure systems can be deployed just by use of free and open source (FOSS) software.

Keywords: SSH security, HTTPS Security, IPtables, Port knocking

1 Introduction

Any system connected to the Internet can be scanned or probed to get detail of any open ports and associated services running on the respective open ports and then to exploit the vulnerabilities of the service to hack into the server. We normally need to connect to the respective service port for different requirements like to manage the system remotely, to remotely access of the data, to remotely access web based email. These open ports are entry point for the attackers. Looking with another angle, if the port is not open, it is difficult to exploit the service. Thus these open ports are the "Achilles heel" of the system.

For example, services like SSH, HTTPS (for this paper it refers to web email access) are running on a predefined well known port like 22, and 443. The default port is always open in the firewall so the service can be accessible over the network. When we use SSH, HTTPS services on standard well known port, we are likely to see brute force login attempts from different or same IP addresses. The SSH, HTTPS does not limit unsuccessful login attempts by itself. There are multiple ways to deal with problem. One good way is to use different non-standard port to hide the service to discover and use the service vulnerabilities to exploit the server. But using latest freely available tools like nmap[1] a complete scan of all the ports on the server will quickly reveal the open port and get the associated service detail like version of the service, running on the server. Then the hackers use the details, to find the vulnerabilities of the service and try to exploit the system using the service vulnerabilities. Another important point is to have a strong authentication mechanism. But this will not stop brute-force, dictionary attacks on the server. The other way is to configure the firewall (IPtables) and allow access of SSH/HTTPS from specific IP Address. However, it is difficult to open specific IP address if the legitimate user access request is coming from dynamically assigned IP address. IPtables can be further customized to stop brute force attacks but still we do not want to make system resources (CPU, Memory of server, and network) busy to deal with unnecessary traffic. Hence, there should be an appropriate security layer in place to protect from such system scanning, and unauthorized access attempts while keeping it accessible online to the trusted users. A good solution is to limit access of SSH, HTTPS services using a technique called - port knocking. The access to the SSH, HTTPS will be unavailable until there is some kind of secret port knock sequence. Then the port will be open for a certain time, and for specific IP address from where the correct port knock sequence observed. It may be relatively difficult for an attacker to find that the remote system uses the port knocking as there is no port open for entering the system. Even the system scan using nmap tool does not show open ports on the system.

Consider that for an attacker who does not know the knock sequence, in order to discover it requires massive brute force effort. That is, without prior knowledge of knock sequence, a simple two TCP port knock sequence (eg 5678, 4567) would require a scan of every combination of two ports in the range of 1-65535. The port would not open until the correct two port knock sequence received. That equates to attempt of a maximum of 65535^2 packets in order to obtain and detect a single successful opening. On the other hand, an authorized user would be able to open the port and access the system from any corner of the world.

In this paper, we have discussed port knocking technique, its implementation using IPtables, our experiment results and proposed a setup using all the available open source tools which adds an additional layer of security to secure SSH, Web email access using HTTPS with very less complexity on the server and client.

1.1 About SSH

The main purpose of SSH [2] is to securely transmit data over network connections using strong encryption and authentication methods. It is a replacement of non-secure Telnet, FTP and r-commands (rlogin, rcp, rsh). Many organizations now use SSH because of its features like secure remote login, secure file transfer, secure remote administration, secure remote-command execution, port forwarding (tunneling).

There are many methods/ways like Replay Attacks, Eavesdropping, Man-in-the-Middle Attacks, IP and DNS spoofing, an attacker might use to gain access of data in transit. SSH mitigate such attacks very effectively.

However, for strengthening SSH, we propose following steps, should be sufficient to protect SSH server and client, even if the number of attacks continues to rise.

(A) Run the service on non-standard port

(B) Defining restricted user access list

(C) Port Knocking [3,4] using IPtables [5]

1.2 About HTTPS

HTTPS (Hypertext Transfer Protocol Secure) is not a protocol in itself, rather it is a security add-on on top of HTTP using SSL/TLS. HTTPS URLs begin with "https://" and use port 443 by default, whereas HTTP URLs begin with "http://" and use port 80 by default. HTTPS is especially important over unencrypted/insecure networks such as Wi-Fi, cybercafé, etc, as anyone on the same network can do packet sniffing and discover sensitive information. Every thing in HTTPS message is encrypted, including headers and response/request.

To prepare a web server to accept HTTPS connection, the administrator must create a public key certificate, also known as the Digital Certificate. Digital Certificates forms the basis of secure HTTPS/SSL session. A certificate is simply a public key containing along with it an identity such as email id, organizations name, URL, It can not only be used for establishing the authenticity of the indentifying entity, but can also be used for encryption (using various key exchange techniques).

A certificate can be self generated and self-signed or a signed certificate can be bought from a Certificate Authority (CA). Both have their merits and demerits.

The organizations allow secure (HTTPS) email access through web based tool like SquirrelMail, TWIG within their own network and also from outside the network. There are many ways to break even HTTPS. Attackers can use vulnerabilities of web application to exploit and hack the server. For hardening web based email access using HTTPS, we propose following steps, should be sufficient to protect HTTPS server which provides web based email access even if the number of attacks continues to rise.

(A) Run the service on non-standard port

(B) Defining restricted user access list

(C) Port Knocking [3,4] using IPtables [5]

For experiment, we have configured latest version of Apache[14], PHP[15], Squirrelmail[13] on CentOS linux platform.

The remainder of this paper is organized as follows. The paper presents a port knocking technology briefly in Section II, followed by Section III presents the details about the implementation and result. Finally, Section IV presents concluding remarks and outlines future work.

2 Port Knocking

Port knocking is a technique that can be used to hide services that are running on a secure and hardened server. This is achieved by not opening the port until a correct sequence of knock packets are received by the server. The client attempts to initiate several three-way-handshakes and receives no reply. These connections attempts are monitored and recorded by firewall (IPtables). Initially, the server presents no open ports to the public and is monitoring all connection attempts. The client initiates connection attempts to the server by sending SYN packets to the ports specified in the knock. It is important to understand that port knocking is an added form of security, and not meant as a replacement for regular security maintenance. This process of knocking is what gives port knocking its name. The server offers no response to the client during the knocking phase, as it "silently" processes the port sequence. When the firewall (IPtables) decodes a valid knock, it opens a port for the specific IP address from where the port was knocked for a specific time. Following figures show how port knocking technique works:

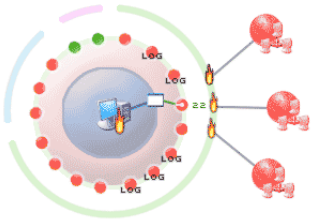

Fig. 1[3]. Default all the ports are blocked

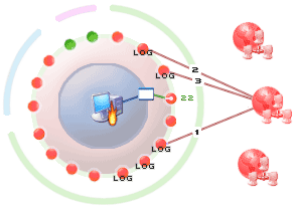

Fig.2[3] User knocks the required ports

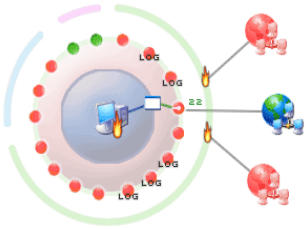

Fig.3[3] The system opens the defined port for specific host from where user has knocked the port

2.1 Pros and Cons of Port Knocking

The main benefit of port knocking is that it allows for stealthy authentication into a host without open ports. The method is stealthy because it is not possible to determine if the host is listening for knocks/requests. Since information is flowing as connection attempts, rather than packet data payload, it is unlikely that this method would be easily detected. The system is flexible, because existing applications such as SSH, which perform their own authentication, do not need to be changed, as port knocking is just an additional outer layer of security for the machine [4]. Consider that for an attacker who do not know the knock sequence, in order to discover it requires massive brute force effort. That is, without prior knowledge of knock sequence, a simple three TCP port knock sequence (eg 5678, 4567, 6789) would require a scan of every combination of three ports in the range of 1-65535. The port would not open until the correct three port knock sequence received. That equates to attempt of a maximum of 65535^3 (over 281 trillion) packets in order to obtain and detect a single successful opening. On the other hand, an authorized user would be able to open the port and access the system from any corner of the world. Modern port knock implementations are much more intelligent and mature, some use highly secure cryptographic hashes in order to defeat the most common attacks like packet sniffing and packet replay.

However, as with any security system, the disadvantage begins with the small inconveniences that must be endured while that system is in use. The use of the client imposes an overhead for each connection and users require instruction. Port knocking cannot be used to protect public services such as mail or web, as this would require everyone to know the knock. Thus, the public services should be relocated to a demilitarized zone (DMZ) and isolated from the hosts with sensitive information.

The implementation of any system that manipulates firewall rules in an automated fashion must be robust to prevent legitimate users and system administrators from being locked out. Also, if the service daemon crashes, the host access will not be available. Appropriate measures should be implemented to avoid such a scenario.

3 Experiment and Result

Our proposed setup solution comprises use of IPtables firewall which is built right into the Linux kernel. However, port knocking can be implemented using various methods. It could be implemented as a standalone daemon processes like knockd, fwknop, etc.. There also exists software to encrypt the entire knock sequence. However, it requires the use of dedicated software on both client and server side for the encryption and decryption process. To overcome these, we have not considered encryption in port knocking. We have configured IPtables with ‘recent module’ which requires initially single port to be knocked. However, it is possible to configure IPtables to sense multiple ports knocking in defined sequence. Hence, there is no specific daemon required to run on the server. Also, client does not require any specific software to knock the port. To avoid denial of service attack, we have also implemented rate control using IPtables.

3.1 Port knocking for SSH (An IPtables based approach):

We have implemented port knocking on the experiment server. We have separately installed latest SSH package. The new SSH is compiled in such a way that it does not display its version in nmap scanning or telnet command line access. This makes it more difficult for the attacker to find the SSH version and use the vulnerability for attacking the server.

To make the system/service more harden, we select the non standard port to run SSH. We have also taken other known security measures to protect the system i.e. allow only protocol version 2, allow only specific users, disable root login, etc.

We have implemented port knocking using IPTables. So, there would be no running service failed that could fail the access. By default, all the traffic is dropped by IPtables rules.

As per our needs, we have configured IPtables rules in such a way that it requires knock only one port and also avoid any extra utility on the client machine for knocking the port. The IPtables rules are written to open SSH port only for the specific IP address from where the port is knocked and keeps the port open for specific time period only. In other words, after knocking the ports, user must establish the session within the specific time period otherwise the IPtables rule drops the request and user should knock the port again to establish the SSH session. We have configured IPtables to log all the request of port scan, port knock and SSH. We have also tried to block some of the common attacks and force SYN packets check, Fragments packets check, XMAS packets check, drop all NULL packets etc.

In order to test the proposed setup, we have kept the system on the internet for more than a year. We have not observed a single unauthorized login attempt which was observed earlier while SSH service was running on well known port number 22. We have also observed that suppose the port is scanned and open for SSH session, at the same time if attacker scan the system using nmap the scan will display scan port either filtered or closed.

To analyze the server logs, we have developed web based software to analyze and display Monthly, Yearly SSH successful and failed login attempts by username, source IP address with IP Address geo-location. The software is developed on Linux using Java, Mysql and Apache-Tomcat. We have used GeoLite[6] freely available database to get geo-location of the IP Address. As mentioned, we have already rejected port 22 access attempt using IPtables. We have statistics of port 22 access attempts with source IP address and respective country. The statistics shows an average of 17 attempts on port 22 per day. Figure 4 shows day wise graphical representation rejected port 22 access attempts for the month of January 2013.

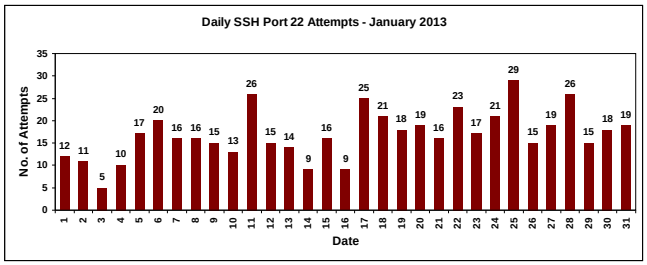

Fig.4 Dropped Port 22 Access Attempts-January 2013

Following table shows dropped port 22 access attempts (January 2013) of top five countries. Figure 5 also represents graphical representation for the same.

Sr. No.	Country	No. of Access Attempts
1.	China	199
2.	US	47
3.	Pvt. IP Address	42
4.	India	34
5.	Korea, Republic	27

Table 1 - Country wise dropped port 22 access attempts

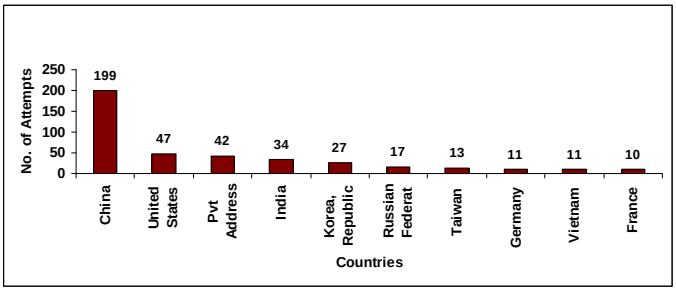

Fig.5 Country wise Dropped Port 22 Access Attempts

In the month of January 2013, there are total 244 number of successful and 18 number of failed (user has entered incorrect password) logins observed.

Sometimes Educational and Research institutes have a requirement to provide the access of Institute subscribed Library Journals/Paper from home/outside network to the students/scientists. To enable the access, we have experimented two mechanisms (1) SSH tunneling and (2) using freely available sshuttel[7] script which we customized as per our experimental need. We did experiment using both the mechanism on windows and linux client. It worked successfully and user could browse Institute subscribed Library Journals/Paper from home/outside Institute's network.

3.1.1 SSH tunneling

Below Figure 6 shows the experimental setup using SSH tunneling to access Institute's network and also to access Institute's Library subscribed journals over Internet.

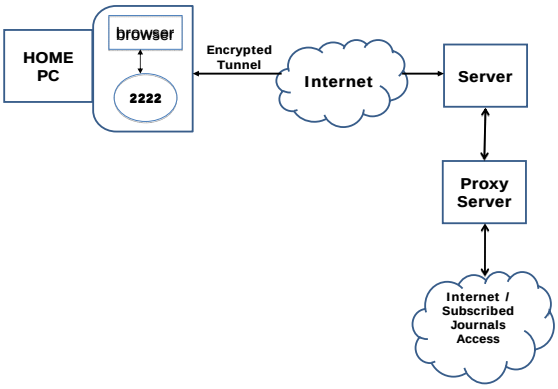

Fig.6 SSH tunneling

3.1.2 sshuttle[7] for Linux:

There is one open source Python based Linux tool/script available called sshuttle to establish SSH based VPN type tunnel. It creates transparent proxy server on local machine for all IP addresses that match 0.0.0.0/0. More specific defined IP addresses can also be used. We have modified the script as per our needs and repacked the whole bundle in a single Linux package file (RPM and DEB) called SSHVPN. Below figure 7 shows experimental setup to securely access Institute's network using sshuttle over Internet.

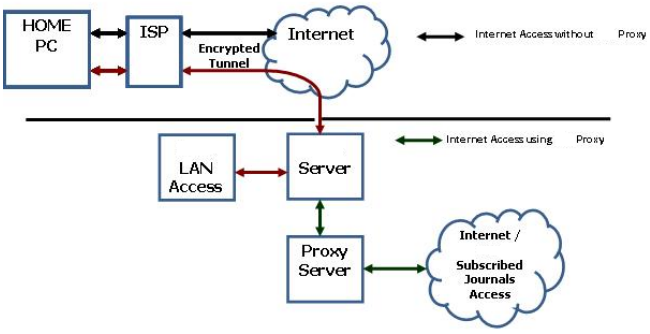

Fig. 7 SSH VPN using sshuttle

We have successfully tested the network access using our customized script from Linux client.

3.2 Port knocking for HTTPS (Web email access):

As explained earlier, the organizations also allow secure email access through web based (HTTPS) within their own network and also outside the network. To further secure from automated brute force password and other automated scripts attacks, we would also like to use port knocking for the web based email access. Now, the problem is that unlike SSH, HTTP is a stateless protocol. As a result, the port has to be kept open for the entire duration of mail access. For experiment, we have configured latest version of Apache[14], PHP[15], Squirrelmail[13] on CentOS linux platform. We have also taken care of basic security of Apache and PHP.

Solution 1:

A simple solution for the problem would be to designate another port which would close the HTTPS port. To automate the close port request, we modified the signout php page of webmail and incorporated the closing port knock. The closing knock port was chosen to be 80 (HTTP port) so that it is not readily visible in browser's address bar.

The solution was extensively tested and was working fine for users accessing mail from their home. However, a new problem arose, if the user was behind a NAT proxy. That is, if more than one user logs in from the same NATed Proxy IP, then the first one to signout will terminate the connection for all users since all 'appear' to come from the same IP address due to Network Address Translation in work behind the scene.

Now, since iptables do not provide a direct way to count the number of requests from same ip address, and take actions accordingly, the only choice we had in deploying this solution was to create a custom program that would monitor iptable's log and would insert or delete rules to allow or deny users. The script would make a note of number of IPs currently connected and would only delete the access rule when the last user from the same IP has quit. The drawback in this approach was that if that external program were to quit or terminate for some reason, then even legitimate users (irrespective of NAT) would be denied of mail access.

Though the problem of NAT would rarely occur, and though the workaround in that case would be to simply allow access for that organization temporarily, the inherent complications involved in dealing with this rare problem, the manual intervention required and dependence on an external program forced us to think of other possibilities.

Solution 2:

We analyzed access logs of the users' web based email access and monitor the duration of their webmail session and found that on average, a user typically spends only around five to fifteen minutes time while checking mail.

Based on that, we chose to close the port automatically after a predefined period of say thirty minutes. So as to not surprise the user by terminating the connection, a count down timer will also be displayed which would show the remaining time till port close. If the user wishes to extend the period, he/she shall again knock the opening port.

If in that duration, another user arrives from the same IP (using NAT) the time period of both the users will get extended.

3.3 Implementation

We have successfully tested both the solutions and found that solution 2 is more suitable without adding a layer of complexity. Since last two months, solution 2 is running on our test server. We have not observed any brute force login attempts, Denial of Service Attacks. We have also configured IPtables in such a way that if someone tries to scan the port, the system will automatically block the IP address for temporary defined time period. However, we are still doing further more experiments.

4 Conclusions and future work

Initially SSH was running on the well-known standard port 22. We have already set the Access Control List (ACL). However, we have observed many unsuccessful login attempts for login id like root, guest, mysql, admin, etc. from different IP sources. Some of the time we have observed unsuccessful brute of attack or Denial of Service attacks.

The multi layer security approach helps us to secure and harden the SSH service. We did experiment on the system with SSH and Port Knocking for more than one year. We have not observed any unsuccessful login attempt and brute force login attacks on the server. SSH with the above setup enable to economically, privately, effectively and safely access the system from public networks like internet.

We have also successfully tested open source - One Time Password (OTP) package on our server. To use this, user need to carry a pre-generated one time password list on a paper or a file. If it is lost/deleted than user can not communicate with the system. Hence, we would like to further study in detail for integration of OTP tool.

Although right now, we are doing experiment using timeout for HTTPS port knock session, in the future, we may instead use the external program based approach if it clears our rigorous testing process. In the near future, IPtables might itself contain branching logic so that we no longer be dependent on the external program.

No single piece of software can be complete security solution. To harden the system, we plan to enable Multi-Factor authentication and also to use available open source fail2ban utility which blocks the IP address for the certain defined time period for the defined unsuccessfully login attempts.

The experimental test setup will be useful to other organization to secure their SSH and web based email access services without any additional complexity on the server and client.

5 Acknowledgements

We thank Dr. A.D.K. Singh, Prof. V.K.B.Kota, Mr. Dholakia G. G., for providing their valuable suggestions and encouragement in establishing this system. We also thank our colleagues Mr. Alok Shrivastava, Mr. Hitendra Mishra, and Mr. Tejas Sarvaiya at the Computer Center and all the PRL users for their support and cooperation.

6 References

[1] NMAP- http://nmap.org

[2] SSH – http://www.openssh.org

[3] Port Knocking – http://www.portknocking.org

[4] M. Krzywinski, “Port Knocking: Network Authentication across Closed Ports”. SysAdmin 2003. Magazine 12: pp 12-17

[5] IPtables – http://www.netfilter.org

[6] GeoLite Database- http://www.maxmind.com/app/geolite

[7] Sshuttle- https://github.com/apenwarr/sshuttle

[8] S. Krivis, “Port Knocking: Helpful or Harmful? – An Exploration of Modern Network Threats”, GIAC Security Essentials Certification, 2004, unpublished

[9] Fwknop – http://www.cipherdyne.org

[10] S. Jeanquier, “An Analysis of Port Knocking and Single Packet Authorization”, MSc Thesis, Information Security Group, Royal Holloway College, University of London, 2006

[11] B. Maddock, Port Knocking: An Overview of Concepts, Issues and Implementations, GIAC Security Essentials Certification, 2004, unpublished

[12] R. deGraaf, C. Aycock M. Jacobson, “Improved Port Knocking with Strong Authentication”. ACSAC 2005, pp. 409-418

[13] SquirrelMail – http://squirrelmail.org

[14] APACHE – http://www.apache.org

[15] PHP – http://php.net

[16] Di Gioia P. , "Behind Closed Doors: An Evaluation of Port Knocking Authentication'. Donald Bren School of Information and Computer Sciences, University of California, Irvine 2004.

[17] Dr. Hussein Al-Bahadili and Dr. Ali H. Hadi "Network Security Using Hybrid Port Knocking" IJCSNS International Journal of Computer Science and Network Security, VOL. 10 No. 8, August 2010