PREVENTING BRUTE FORCE ATTACK THROUGH THE ANALYZING LOG

Bahaa Qasim M. AL-Musawi
bahaaqm@yahoo.com
Department of Electrical Engineering, College of Engineering, University Of Kufa . Kufa-Iraq

Abstract

Secure Shell (SSH) is a secure remote login program which can be used in place of regular telnet. It has become the default remote access method for administration of UNIX systems. It is very common for public Internet facing servers to experience attacks that attempt to brute force username and password combinations via SSH to gain access.

This paper examines these attacks depending on SSH log file to find unsuccessful logins then blocks IP addresses of unsuccessful logins for a period of time that is decided by administrator and then send an e-mail to administrator to consider whether the addresses blocked belong to users failed to access or by an attacker, finally the administrator will block attacker's IP address forever.

Some attackers highly skilled and just used trusted IP address as a user name then the software will block the IP address of attacker as well as the victim IP address that is used by attacker. In this paper, an adaptive mechanism was built-in to distinguish between attacker IP address and victim IP address which may be used by an attacker, and then the program will block just attacker IP address.

Keywords: IPtables Firewall, SSH, SSH application

منع القوة الغاشمة من خلال تحليل السجل

بهاء قاسم محمد الموسوي
قسم الهندسة الكهربائية، كلية الهندسة، جامعة الكوفة. الكوفة-العراق

الخلاصة

القشرة الآمنة (The secure shell) هو برنامج لتامين الدخول عن بعد والذي يمكن استخدامه بديلاً عن بروتوكول تل نيت (telnet). لقد اصبح بروتوكول القشرة الآمنة هو البروتوكول الافتراضي للوصول عن بعد لادارة انظمة يونيكس. انه من الشائع جداً ان تتعرض خوادم الانترنيت المنتشرة في العالم لهجمات القوة الغاشمة التي تحاول سرقة اسم المستخدم وكلمة المرور عبر استخدام بروتوكولات القشرة الآمنة من اجل الولوج الى الخوادم. هذا البحث يمنع هذه الهجمات من خلال الاعتماد على ملف السجل (log file) الذي يحتوي على تسجيلات الدخول لبروتوكول القشرة الآمنة ومن ثم يتم حجب عناوين بروتوكولات الانترنيت لاي عمليات ولوج غير ناجحة لفترة من الزمن يتم تحديدها من قبل المسؤول عن الخوادم ثم يتم ارسال بريد الكتروني الى المسؤول للنظر فيما اذا كانت العناوين المحجوبة من قبل مستخدمين موثوقين فشلوا في الولوج او من قبل بعض المتطفلين حيث يمكن للمسؤول حجب عناوين المتطفلين بصورة نهائية في حال تكرار محاولاتهم. بعض المهاجمين من ذوي الخبرات العالية يستخدمون بعض العناوين الموثوقة كاسم مستخدم عندها البرنامج سيقوم بحجب عنوان المتطفل بالاضافة الى عنوان الضحية الذي يمكن ان يستخدمه المتطفل كاسم دخول. في هذا البحث تم وضع آلية للتمييز بين عنوان المتطفل وعنوان الضحية الذي ان يستخدمه المتطفل لتكون عملية الحجب لعنوان المتطفل فقط.

الكلمات المفتاحية: جدار الحماية لجدول عناوين الانترنيت، القشرة الامنة ، تطبيقات القشرة الامنة

1. Introduction

The Secure Shell (SSH) protocol is one of the most widely used mechanisms designed to secure many of the activities that are carried out on the Internet. It is used to secure remote shell operations by means of cryptographic authentication mechanisms [1]. Furthermore, it can protect the privacy (through strong cryptography) not only of remote shells and file copying, but of generic application-layer flows through what is known as "port forwarding" [2].

In general, the privacy of SSH users can be seen as protected on two fronts: the contents of actual data that flows through an SSH session is supposed to be impenetrable by a third-party observer. At the same time, the type of activity that the user is carrying out through the SSH connection, it could be web browsing, secure copy, remote shell execution, etc., is supposed to remain private [3,4].

There are two fundamental approaches to attacking SSH, exploits and brute force. Exploiting security issues in old versions of SSH is typically used by worms. While the brute force attack is typically a manually initiated process used to scan large ranges of IP addresses to find vulnerable accounts [5].

The paper organized as follows: section 2 present a survey of brute force attack. Section 3 illustrates SSH applications. Section 4 describes IPtable. Section 5 presents the implementation of the algorithm and finally, section6 presents the conclusions of this work.

2.Related Work

There have been numerous studies that detail network monitoring, security, and SSH attacks [6, 7, 8]. Thames et al. outlines a distributed architecture for preventing SSH attacks [9]. In this architecture, a group of trusted hosts works together and shares information in order to block attackers and modify firewall rules. The architecture was never implemented, only simulated. A similar, lightweight option is Denyhosts. Denyhosts is an open-source Python script that maintains a blacklist based on past failed login attempts [10]. Review the current market for brute-force attacks against SSH-related software body, such as: denyhosts, sshdfilter and sshit [10,11]. The common point is that both use the log file for SSH server that is located on /var/log/auth.log, then analyze the log file to block IP address of failed login attempts.

Some attackers highly skilled just use trusted IP address as a username then the software will block the attacker IP address as well as the trusted IP address that is used by attacker.

3. SSH Applications.

There are many applications for SSH that can be used from Unix/Linux command line. With SSH applications, anything that can be accomplished at a machine's command prompt can now be done securely as described below:

- **Use for remote login**

The syntax for remote login in Linux system is: [$ ssh <username>@<remotehost>] Where remote host may be domain or IP address [12].

- **Typical syntax for remote command execution**

It possible to execute commands in remote host with one line command instead of login and then execute typical commands, for example, who command which lists all users logged on to the remote host may be used [13].

- **Forward X Connections**

To use a graphical program that runs on a remote host on the local machine, the -X option must be set. This can be done for running a single program: [$ssh -X <username>@<remotehost> <graphical programs name>] [12,13]. ·

- **SSH Port Forwarding**

SSH has the functionality of forwarding ports between the client and the server through the encrypted connection. This is working with the standard SSH implementations, no extra software is needed. This is also called "encapsulation" or "tunneling". There are forward and backward tunnels possible [12,13].

A. Forward Tunnel

The syntax of forward tunnel is as follows: [ssh -L < local_port >: < host_to_forward_to >:< remote_port >user@sshserver].

This will open a port (local_port) on the ssh client host. All connections to this port will be forwarded (tunneled) via the ssh connection to the remote host (ssh daemon).

As an example, we have a situation, where a web server on port 80 is running on a host (Host A) and a firewall is blocking connections from outside to this host and port, but allowing ssh connections on port 22. To access the web server from the outside (Host B), an ssh connection with a port forwarding can be established: ssh -L 2080: localhost :80 user@sshserver

As long as this ssh connection remains established, all connections to the port 2080 on

the client side (where ssh has been started, Host B) will be forwarded to the port 80 on the "ssh server" (Host A) host. In this case, one can type http://localhost:2080 into a web browser on the client (Host B) and be directed to the web server (Host B) via the encrypted ssh connection.
Another scenario is an encrypted connection to services that are unencrypted, or when the connection is restricted to the localhost only [14].

B. Reverse Tunnel

The syntax for a reverse tunnel is: ssh -R <remote_port>:<host_to_forward_to>:< local_port> user@sshserver
As with the forward tunnel, a port will be opened for listening, but here on the remote host (ssh server). All connections to this port will be forwarded back to the ssh client host [14].

Secure copy SSH can be used for secure copy between two hosts, for example: $ scp <username>@<remotehost>:<sourcepath>/ <destinationpath>
Copies a file from a remote host to the local machine $ scp <sourcepath> / <username>@<remotehost>: <destinationpath>
Copies a file from the local machine to a remote host [12].

4. IPtables

Iptables is part of the Netfilter project. Netfilter is a set of Linux kernel hooks that communicate with the network stack. Iptables is a command and the table structure that contains the rule sets that control the packet filtering.
Iptables is complex. It filters packets by the fields in IP, TCP, UDP, and ICMP packet headers. A number of different actions can be taken on each packet, so the key to iptables happiness is simplicity. Start with the minimum necessary to get the job done, then add rules as you need them. It's not necessary to build vast iptables edifices, and in fact, it's a bad idea, as it makes it difficult to maintain, and will hurt performance [15, 16].
There are three tables in iptables. Any rules or custom chains that you create will go into one of these tables. The filter table is the default, and is the one most used. The filter table contains these built-in chains:
INPUT: Processes incoming packets FORWARD: Processes packets routed through the host OUTPUT: Processes outgoing packets

5. Implementation

The idea of the model started after checking log file for SSH server through analyzing SSH log file looking for any login attempt failed, the checking process produced hundreds of failed attempts by attackers to server starting by using root access and ending using dictionary of well-known username/password combinations.
In this paper, a mechanism that checks log file for failed attempts and block their IP addresses for a while and then send the blocked IP addresses to the administrator using SSMTP or secure simple mail transfer protocol, *ssmtp* is a send-only sendmail emulator for machines which normally pick their mail up from a centralized mailhub (via pop or imap) [17].
Some attackers may use an IP address as a username, and then the blocking will be done for the IP of the attacker as well as the IP of victim that the attacker may use. Then the attacker will fail in attempting to access but will succeed in blocking specific IP address. The adaptive model that is built-in in this paper is capable of distinguishing between attackers' IP and the IP of the victim which attacker want to block it as well be soon explained. The model was tested and built on Debian 5.0 server that provide DNS serves and web server over more than 100 PCs connected on internet.
A typical SSH brute force will be logged by SSH via syslog using the auth-priority. Depending on the SSH software, typical brute force log entries consist of the following:

^ Sep 2 18:37:59 DNS1 sshd[828]: Failed password for invalid user admin from X. Y.Z.W port 52273 ssh2
^ Sep 2 18:38:08 DNS1 sshd[830]: Failed password for invalid user user from X.Y.Z.W port 52320 ssh2
^ Sep 2 18:38:16 DNS1 sshd[832]: Failed password for root from X. Y.Z.W port 52377 ssh2
^ Sep 2 18:38:25 DNS1 sshd[834]: Failed password for invalid user admin from X. Y.Z.W port 52433 ssh2

These were four login attempts, two for non-existing accounts called user and admin, and the third for the root account.
The above scenario can be stopped by denyhosts, sshdfilter, sshit as well as our model, but if an attacker use IP address for a victim as a username then the blocking will be for attacker's IP address as well as IP of the victim as shown bellow.
^ Sep 1 21:48:39 DNS1 sshd[854]: Failed password for 8.8.8.8 from 212.6.40.14 port

58455 ssh2 ^ Sep 2 09:20:54 DNS1 sshd[844]: Failed password for 8.8.8.8 from 222.168.102.78 port 51961 ssh2 The filtering process will be as follows: 8.8.8.8 X.Y.Z.W 8.8.8.8 X.Y.Z.W

The model is capable of distinguish between attacker IP address and IP address of victim that may be used by the attacker as shown in (figure 1). (Figure 1) shows the model and how it filters log file to find failed passwords and its IP address to block it as well as it is capable to distinguish between attacker's IP address and victim's IP address that may be used by the attacker.

Filtering process will produce the IP addresses of just failed password as follow: X.Y.Z.W

So, process of checking each line of failed attempt to find out if the attacker uses an IP address as a username is necessary.

After building the module and testing it, there are three options to make this module work at each reboot of server if server shuts down:

1. Using cron daemon to make this module working as each start for server.
2. Adding the module to System V Init. There are seven run levels in Unix OS, a module can be added to which run level you wanted.

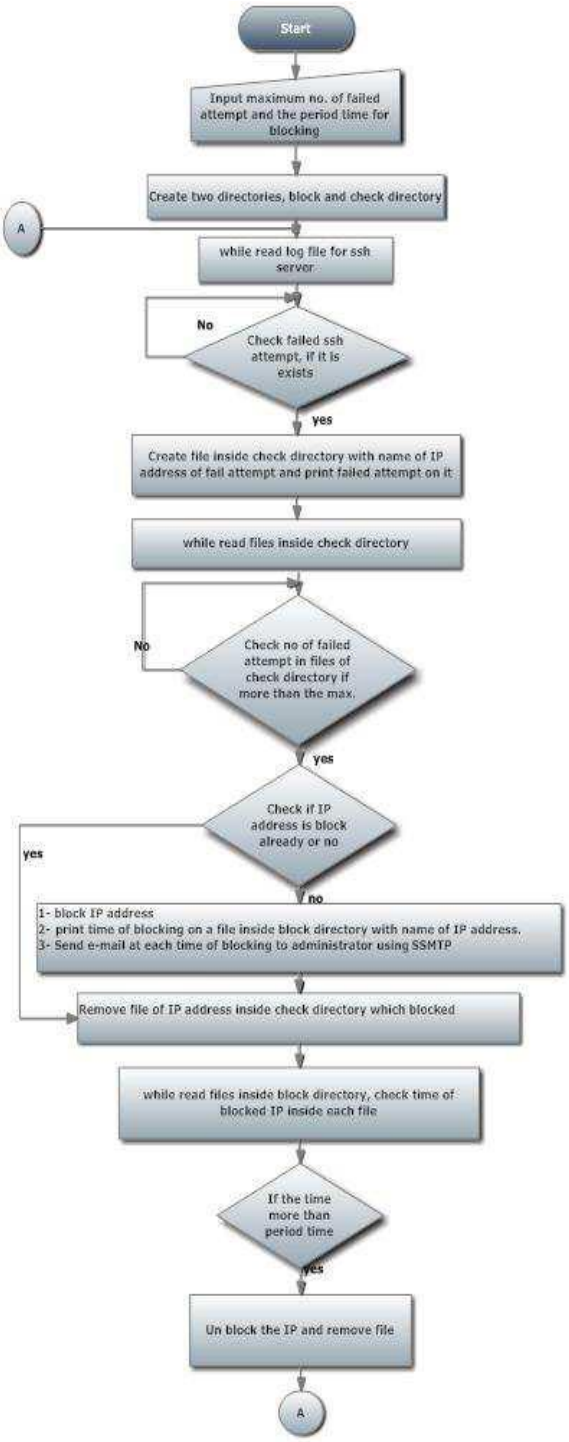

Figure 1: Flowchart of checking and blocking process

3. Adding the module to Upstart service. To make the module working after starting networking can be configured using upstart service.

Cron daemon was adopted in this work.

Because the server used is necessary to work all the time and if it stops for any reason and starts again, cron daemon will start the program automatically. This can be done by typing crontab –e then add @reboot <the path to script>. To download the script, please visit : http://www.kuiraq.com/staff/bahaaqm/software

6. Conclusions

In this paper, a compact model with real-time detection to prevent SSH brute force attacks was presented. This model blocks IP addresses of unsuccessful logins for a period of time that is decided by administrator, the model gives remote monitoring to administrator about who try to hack the server through sending e-mails about the blocking IP address.

After installed the model for more than three months, it's note that some attackers highly skilled and just used trusted IP address as a username then the software will block the IP address of attacker as well as the victim IP address that is used by attacker, this issue is a problem of denyhosts and sshit. This issue was noticed by the remote monitoring and notice that there are many of trusted IP addresses was blocked, so an adaptive mechanism was built-in to distinguish between attacker IP address and victim IP address which may be used by an attacker, and then the program will block just attacker IP address.

References

1. Liberatore, M. and Levine, B. N. **2006** "Inferring the source of encrypted http connections+", in CCS '06: Proceedings of the 13th ACM conference on Computer and Communications Security, (Alexandria, Virginia, USA), pp. 255–263.
2. Ramsbrock, D.; Berthier, R. and Cukier, M. **2007**. "Profiling Attacker Behavior Following *SSH Compromises*", in Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp.119- 124.
3. Dusi, M.; Gringoli, F. and Salgarelli, L. **2008** "A Preliminary Look at the Privacy of SSH Tunnels", in Proceedings of the 17th IEEE International Conference on Computer Communications and Networks (ICCCN 2008), (St. Thomas, U.S. Virgin Islands), Aug.. Open SSH Security, http://ww.openssh.org/security.html.
4. Dusi M.; Gringoli F., and Salgarelli, L. **2008.** "A Preliminary Look at the Privacy of SSH Tunnels", in Proceedings of the 17th IEEE International Conference on Computer Communications and Networks (ICCCN 2008), (St. Thomas, U.S. Virgin Islands), Aug.
5. Mukosaka, S. and Koike, H. **2007** "Integrated visualization system for monitoring security in largescale local area network", Asia-Pacific Symposium on Visualization, **0**:41–44,. http://doi.ieeecomputersociety.org/10.1109/APVIS. 2007.329273.
6. Anirudh, R. and Nick, F. **2006,** "Understanding the network-level behavior of spammers" , in SIGCOMM '06: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communications, pages 291–302, New York, NY, USA, 2006. ACM. ISBN 1-59593-308-5. http://doi.acm.org/10.1145/1159913.1159947.
7. Daniel, R.; Robin, B., and Michel, C.,**2007.** "Profiling attacker behavior following ssh compromises", dependable Systems and Networks, International Conference on, **0**:119–124.