International Journal of Performability Engineering, Vol. 5, No. 1, 2009, pp. 55-70.

Printed in India

Rebound Wall: A Novel Technology against DoS Attacks

YUAN-SHUN DAI[1,2*], XUEPING LI[1], XUKAI ZOU[3] LIUDONG XING[4]

[1] *Department of Industrial & Information Engineering, University of Tennessee, USA,*
[2] *Department of Electrical Engineering & Computer Science, Univ. of Tennessee, USA*
[3] *Department of Computer & Information Science, Purdue University, Indianapolis, USA*
[4]*Dept.of Electrical & Computer Engineering, Univ. of Massachusetts, Dartmouth, USA.*

(Received on Dec.19, 2007)

Abstract: DoS/DDoS attacks have become one of the most critical security problems in today's network systems, which is easy to launch by hackers but hard to protect by victims. This paper presents a novel and robust mechanism, named Rebound Wall, which proves very effective to protect a victim server from DoS attacks and easy to deploy in practice. The rebound wall comprises of available machines in the LAN, surrounding the core server. Unlike the existing DoS defense techniques which rely much on marking and/or filtering, the rebound wall utilizes roaming crypt-doors. Valid requests can only go through a designated entrance to the server. These entrance machines are roaming over the rebound wall, so that hackers cannot find the target to launch effective attacks. Some other new technologies and protocols that are necessary to furnish the rebound wall technology are also presented in this paper, including Floating Entrance, Entrance Switch, User-end Authentication, Entrance-based Privilege Control, and Traceback. A survivability model is further built for the rebound wall based on a CTMC. A rebound wall was implemented in reality. Both experimental data and analytical results validated the effectiveness, efficiency, and robustness of the rebound wall technology. We finally compare the rebound wall with other related and advanced technologies against DoS/DDoS.

Keywords: Security, DoS (Denial of Service), Survivability, Modeling, Markov Process

1. Introduction

The dependability and security have become critical issues in today's network systems, because the Internet and a large number of networks are not dependability/security-oriented when designed, see *e.g.,* [21] and [19]. The hackers and errors (including human errors) are constantly compromising those vulnerable holes residing in hardware, software or systems to launch attacks and cause failures [1], [2], [4] and [5].

One of the most critical issues with respect to the Active attacks is the DoS (Denial of Service). DoS conquers a computer or network by consuming the computer/network resources that would otherwise be used to serve legitimate users [13]. There are typically two classes of DoS attacks: logic attacks and resource attacks. Logic attacks, such as the "Ping-of-Death", exploit the flaws existing in victim's software/system to crash remote servers or substantially degrade the servers' performance. The resource attacks overwhelm the victim (*i.e.*, Request buffers, CPU, memory, or network resources) by pouring a large

*Corresponding author's email: ydai1@eecs.utk.edu

number of spurious requests. For example, the best known DoS attack is the "SYN flood", in which the attacker sends a stream of TCP SYN packets to exhaust the victim's TCP connections. DoS attacks are easy to launch but extremely hard to defend because there is no simple way to distinguish the "good" requests from the "bad" ones. Moreover, the DoS attackers often forge the IP source address (*i.e.,* IP Spoof) of each packet they send. Consequently, it is also difficult to determine the real sources of the packets (unless the routers mark the packets passing though them). One powerful format of DoS attacks is called Distributed DoS attacks (DDoS).

Many methods have been suggested to defend against the DoS or DDoS attacks. The defending methods can be implemented on a single node, such as the protected server, or on multiple collaborating nodes. The single-node defending methods, see *e.g.,* [17], [8] and [3], can observe attacking symptoms when the traffic to the victim accumulates to a warning level. They launch the traffic filtering to protect the victim subsequently. The hop-count filtering [8] was a solution relying on the fact that the number of hops between the attacking source and the victim was unchanged within a short period. The traffic level measurement [3] protected a victim by dropping most incoming packets when total incoming traffic shooted to a high level. This measurement can identify attack sources by using statistical property of observed traffic flows. SYN flooding attacks can be observed at leaf routers that connect end hosts to the Internet [17]. The attack detection was based on the fact that in normal network traffic the SYN and FIN pair should appear symmetrically. To accumulate these pairs, they used a non-parameter CUSUM method that needed to estimate the average duration time of TCP connections, which varied according to a network's status. These single-node defending methods, however, inevitably cause collateral damage on legitimate traffic destined for the victim that the system is trying to protect. This paper attempts to minimize the influence on legitimate users.

The multiple-node defending methods require nodes (*e.g.*, routers and servers) distributed in networks to cooperate in preventing DoS attacks. Multiple routers can mark packets passing through them, see *e.g.*, [20] and [14]. As a result, the packets carried the path information, so it was feasible to detect packets with IP address spoofing and traced back to real attacking source. Filtering schemes can be activated at the closest router to the attacking source to filter out illegitimate traffic and stop possible attacks. Intermediate routers can start to drop packets to a destination when the traffic reached a certain threshold on a link relevant to the destination, as presented by [7]. The authors in [15] presented DefCOM that consisted of heterogeneous defensive nodes to detect and restrain attacking traffics. Three types of defensive nodes, *alert* generator nodes, *core* nodes and *classifier* nodes, were distributed in a peer-to-peer network and they must communicate frequently to achieve dynamic cooperative defense. Although these methods provide secure and efficient countermeasures to DDoS attacks, they require the modification and cooperation on routers and thus cause difficulties for deployment. Such intensive communications or expensive overhead are mainly consumed by the filtering functions in an intelligent manner to detect the DoS attacks and differentiate them from the legitimate traffics. The novel technology proposed in this paper does not need the complicated filtering mechanisms, but rather utilizes the human characteristics of known and unknown factor, *i.e.,* the legitimate users know and can recognize the roaming entrance but the DoS attackers never have chance to know it. This is more effective without much overhead than the real-time filtering. The honeypot technology proposed recently provides another

perspective to detect and prevent the DoS attacks in which a honeypot is a target to attract hackers to analyze ongoing attacking activities, see *e.g.*, [16], [11] and [10].

This paper presents a novel technology, called rebound wall, which proves very effective to defend against the DoS or DDoS. It is easy to implement and deploy in practice. The core server that is offering services does not directly receive the requests from the external users, but is protected by a rebound wall comprising of a set of available machines in the LAN. One designated machine, called as entrance, receives valid requests from authorized users and transfers the requests to the server. These entrance machines are roaming over the rebound wall. As a result, illegitimate hackers will lose the target for attack while valid users can get the correct entrance information according to the newly proposed mechanisms: user-end authentication protocol and entrance-based privilege control. The machines in the rebound wall consistently monitor the entrance machine and can efficiently detect problems of the entrance machine and entrance switch can immediately recover the services. The rebound wall also introduces a portable key scheme which not only increases the flexibility and portability for users, but also improves security levels for authentication. Moreover, the rebound-wall technology endows a strong capability for intrusion detection by simply checking wrong knocks on the rebound wall.

Thus, the novel rebound wall technology offers a simple, straightforward, and cost-effective option to defend against the DoS/DDoS attacks. The rest of the paper is organized as follows: section 2 presents the architecture of the rebound wall and other new technologies and protocols that are necessary to furnish the rebound wall technology; section 3 develops a Markov model for the rebound wall system and measures the survivability that can reflect the capability of a rebound wall to defend against the DoS/DDoS attacks; The implementation and analysis of the rebound wall is shown in section 4; Finally, section 5 concludes this paper and discusses some future extensions.

2. Rebound Wall

A novel mechanism, named Rebound Wall, is presented here to protect a core server from external DoS attacks. The necessary components of the rebound wall include: Floating Entrance, Entrance Switch, Authentication, Privilege Control, Secure Entrance Information Distribution Protocol, and Traceback. These components are depicted in Fig. 1 and will be elaborated in the following subsections, respectively.

2.1 Floating Entrance

A core server that provides services locates in a certain LAN where there are other PCs that can form a rebound wall to protect the core server from external DoS attacks.

The core server does not directly receive the requests from the external users, but only from one designated machine in the LAN. The designated machine is randomly selected by the core server from all candidate machines that form the Rebound Wall. The task of the designated machine is to receive, analyze and transfer the requests to the core server. At anytime, only one (or several) designated machine(s), called as the *entrance(s)* of the Rebound Wall, receives valid requests from authorized users, as depicted by Fig. 1. However, the users do not know which machine (IP address and Port Number) is designated at the current time to accept the service requests, because the entrance is floating over the Rebound Wall (*i.e.,* periodically change to another machine). Such Floating Entrance is an important character to protect the core server from DoS.

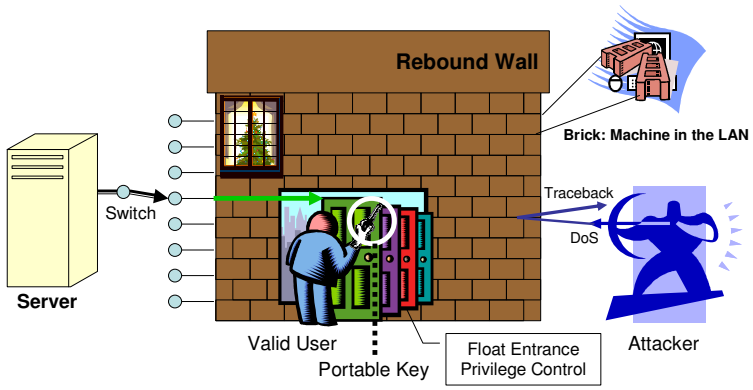

Figure 1: The general description of the rebound wall.

Once the core server designates an Entrance machine, it will let the whole Rebound Wall know this Entrance Machine (including the IP address and Port Number). At the User-End, there is a buffer that stores the last one (or several) entrance(s) (*i.e.,* their IPs and the port numbers). When a legitimate user logs in before requesting services, the user-end application will automatically talk with the last entrance machine. The old entrance will tell the user about the current entrance through a protocol that will be discussed in later subsection 2.4. The user-end application then updates its buffer with the current entrance information. After the user passes the authentication at the current entrance, the requests for services or for connection can be forwarded to the core server through the entrance on the rebound wall. The entrance information sent to users is encrypted according to a new and special protocol, which is called Secure Entrance Information Distribution and will be discussed in later subsection 2.4.

In addition, the buffer at the user-end can store multiple previous entrances. If one is not available due to failures or jamming, then the next one in the buffer list is tried. Also, a machine on the rebound wall may open multiple ports to receive the requests for entrance information in order to guarantee all legitimate users' accessibility, as the following mechanism. When a machine (along with its one port) is designated as the entrance, the machine builds a user list for this port and records those users who have used it. Then, when the entrance is switched to other machines or other ports, this port serves for providing the new entrance information to those users who may request in future. After one user asks for the new entrance, his name is removed from the list of this port. When the list becomes empty, this port can be totally closed. With this scheme, each machine on the rebound wall may open multiple ports simultaneously, which as a byproduct can further confuse the hackers who are unable to observe the real entrance.

Under this scheme, the hackers (or invalid users) do not know the correct entrance, so when they use DoS to attack the core server, the packets cannot go though the wall via the correct entrance and will be rebounded. When a large number of requests from DoS attacks reach a wrong machine that is not currently designated as the entrance, such illegitimate intrusions can be easily detected under our rebound wall mechanism. Then, the traceback is triggered to find who/where the origin to generate this DoS attack is? which is out of the scope of this paper. In case hackers guess the entrance machine's IP address correctly by chance, it cannot get the correct Port Number so that such intrusion can also

be detected when requests arrive at a wrong port or at all (multiple) ports. If the hackers simultaneously send a bunch of requests to all machines and all ports of the whole LAN, the Gateway of the domain will not allow such bunch of requests to pass, because it is too obvious that such kind of requests is a DoS attack and is easy to be prohibited by the Firewall of the Gateway that governs the LAN.

2.2 Switch Entrance under Failures

The rebound wall scheme has very high survivability under the DoS attacks, as will be analyzed in section 3 for more details. Let's suppose, though the chance is almost negligible, the hackers are lucky enough to guess both IP address and port number of the entrance machine correctly. However, the DoS attacks can only make this entrance machine unavailable by blocking the buffer for requests before authentication. It is because without a valid PIN or key from a hacker, the requests will not be forwarded to the core server, as depicted by Fig. 1.

The other machines of the rebound wall have another function to monitor the entrance machine (such as periodically send a Pseudo-Request to the entrance to see whether it works or not). If any other machine finds the entrance machine is not properly working without correct response, it will report to the core server and thus the core server can designate another machine as the entrance of the rebound wall to continue the service. In the meantime, the intrusion has also been detected due to the unavailability of the prior entrance machine. Thereafter, the traceback can be triggered and this unavailable machine can be recovered (such as discard all requests in the buffer or reboot) without affecting the core server or major service. In fact, we can also launch an active method to denote the core server from a jamming Entrance machine when the outside requirements are close to its ability (such as a predefined threshold of the entrance buffer). Thus, whenever an attack happens and one Entrance machine is out of service, the switch can be made immediately.

This rebound wall together with the entrance switching scheme can tolerate not only the external hackers' attacks but also the internal users' attacks. Since the internal users know the entrance, they can easily initiate the DoS attacks if they have a malicious purpose. Under our rebound wall scheme, they cannot know the updated entrance information until they log in and pass the authentication by the previous entrance, so their real identifications (User ID or PIN) can be captured. If their attacks are detected via the above mechanism, the user accounts will be blocked (blacklisted) by the core server and further investigated before reactivated.

If we shut down all services provided by an Entrance machine that is going to be idle because it is under attacks, the services to valid customers could be affected too. We may construct a safe list to contain some IPs or users such that if those IPs/users are using services from the core server, their services will not be interrupted because we can switch the connection from one Entrance machine to another opening Entrance machine seamlessly. Other unsafe traffic requirements in the old Entrance machine could be simply ignored to guarantee the healthy state of the newly opening Entrance machine.

2.3 Authentication and Entrance-based Privilege Control

To authenticate the users and prohibit eavesdropping, we hereby present a new protocol for Authentication and Data transmission based on cryptography. This can also realize the privilege control via different entrances.

Authentication

At the first step, the authentication is an important issue to decide whether the wall should let the requester know the real entrance. If we implement the user account and password (or PIN), though simple, it needs that each window (machine) in the rebound wall should have a database to record every user's login information. However, this is not the most cost-effective and thus increases the overhead for synchronization. Another alternative can be that only the core server prepares the database, and then the rebound wall machines can communicate with the core server for authentication on each request. Although this method is more economic with less storage, yet the DoS attacks may seek another way to compromise the system because each request has to be verified by the core server.

Therefore, we present a novel protocol for the first level authentication when requesting entrance from a rebound wall, which is named as the cryptography-based authentication at user end. The basic idea is that the IP and Port Number will be hidden/encrypted in a public message sent to the requester. Each valid user should have held a key that was assigned when they registered. Therefore, the authentication can be made at the user-end not at the rebound wall to solve the problem for eliminating the DoS risks. Those valid users can decrypt the correct IP and Port from the received message while the external hackers who do not have a valid key cannot get the correct information though they may also receive or intercept the message. The detailed protocol will be presented in the following subsection 2.4.

Entrance-based Privilege control

There are different levels among various users. For instance, some trial users may be free of charge for the service while some other VIP users may pay the money for better services. Most existing mechanism for privilege control is mainly based on users' login information. Here, by the rebound wall, we propose a novel privilege control called as Entrance-based privilege control.

The idea is that the rebound wall opens multiple entrances for different users with corresponding privileges. For example, a user at a lower level only knows the entrance with lower privilege while a user at higher level knows higher-privilege entrance. Thus, the requests forwarded from different entrances to the core servers will be treated differently. For example, the priority of the requests from a higher-privilege entrance is superior to those from a lower-privilege entrance so that they can be served first or assigned more resources on them (such as computational resources, storage resources, or network bandwidth etc.). Some other special operations for higher-privilege users can also be differentiated through the entrances, *e.g.*, some requests are allowed to pass through certain entrances while some others are not allowed to pass due to the privilege limit. Therefore, the core server does not need to care about the privilege control which is also expensive with much overhead because the entrances have filtered the requests based on the corresponding privileges assigned by the core server initially (such as what operations/requests are allowed and what are not etc).

However, the challenge here is how to differentiate users with corresponding privileges in a cost-effective manner so that they can go to the correct entrance without exceeding their authority to other entrances. The following subsection 2.4 presents a novel protocol which realizes the functions of user-end authentication and privilege control.

2.4 A Novel Protocol for Entrance Access Control

The protocol consists of two functions: the first function is to solve the problem of user-end authentication; the other enables the privilege entrances control, which is built upon the former function.

It is assumed that every valid user in the system is assigned a secret group key, denoted by SID_i for a group of users denoted by U_i. The group means that those users who share the same privilege. In the registration process of a user, the secret group key can be transmitted through a secure channel to the user (such as RSA). Assume P is a large prime which forms a finite field F_p. Whenever the machines in the rebound wall respond the information of entrance's IP and Port, a polynomial $A(x)$ is constituted by:

$$A(x) = \prod_{i \in \psi} (x - f(SID_i, z)) \tag{1}$$

where ψ denotes those groups who have the privilege to know this entrance. Here, SID_i s are secret keys assigned to the groups contained by ψ. $f(x, y)$ is a public one-way function and z is a random integer from the field F_p. $A(x)$ is called as *Entrance Control Polynomial* (ECP). As Eq. (1), it is obvious that $A(x)$ is equated to 0 when x is substituted by $f(SID_i, z)$ from a valid user whose SID_i belongs to the set ψ; otherwise, $A(x)$ is a random value if other unauthorized numbers are plugged.

Then, the rebound wall machines generate an integer $K = IP|Port$ which simply appends the 4-digit port number to the 12-digit IP address. The rest higher digits are all set as 0, because usually the size of a big integer for a cryptosystem should be 128 bits at least (about 30 digits). Then, *K* is hidden or encrypted in the following polynomial

$$P(x) = A(x) + K. \tag{2}$$

Finally, the rebound wall machines send $(z, P(x))$ to the users.

From the received information that hides the entrance information, any valid group member U_i can get the entrance information by

$$K = P(f(SID_i, z)). \tag{3}$$

Here the user computes $f(SID_i, z)$ first and then substitutes the result in $P(x)$. Thus, the user knows the entrance of the rebound wall as the IP is the first 12 digits and the port number is the last 4 digits of *K* after ignoring the 0s at the higher digits.

Any other member U_r without sufficient privilege to use that entrance is not included in ψ, so $P(f(SID_r, z))$ yields a random value instead of *K,* where U_r cannot know the hidden entrance. The external hackers without any valid keys are similarly unable to get the correct entrance information. This protocol guarantees that only a user whose SID_i (*i.e.*, $f(SID_i, z)$) is included in ECP $A(x)$ can extract the entrance information from the polynomial. Therefore, the authentication is realized at the user-end by key.

For privilege entrance control, the following steps can guarantee all users to reach the entrances for their corresponding privilege:

1) Suppose M entrances are opened now with M different privileges each associated with a set ψ_i (i=1,2,…M) for the groups of users to use entrance i. Without losing generality, we suppose the highest privilege entrance is entrance 1, and the privileges decrease along with the increasing of the entrance numbers till entrance M that has the lowest privilege.
2) The rebound wall generates a set of polynomials in sequence from the highest privilege to the lowest one, $\{(z_1, P_1(x)); (z_2, P_2(x)); ...; (z_M, P_M(x))\}$ to hide all entrances by $\{K_1, K_2, ..., K_M\}$, and sends to all users.
3) Each user can decrypt a list of entrances from the received list of polynomials. Some of the decrypted messages may not be correct entrances due to the mismatched privilege of the user. Anyway, the user can attempt to connect the entrances from the first one successively until the one that is succeeded in connecting to the core server through the correct entrance. Thus, the user reaches the correct entrance associated with his privilege. In fact, most K_i s, if incorrectly decrypted, are obvious because they are random numbers which do not exactly have all higher digits be 0s except the last 16 digits, different from the original format of K incorporating the entrance $IP|Port$.

2.5 Portable Key Scheme

After designing the authentication and entrance-based privilege control mechanism for the rebound wall, the personal secret key becomes another important ID for an authorized user. We hereby propose a portable key scheme which not only can assist the rebound wall system against DoS attacks but also can help increase the security level for authentication.

The portable key scheme is based on a key file generated when a user first registers. The registration is processed before the user starts using the service system. The key file that contains a personal secret will be sent to the user through a secure channel (such as the two-party communication by RSA). The key file also includes an initial entrance that is opened at the same time as the registration. The user can store the obtained key file in any storage media (such as USB flash drive, soft disks, etc.).

Then, the users can request the service at any place with the portable key file. When a user logs in, he should not only enter his PIN but also insert his storage media with the key file. This is a more secure authentication process than only entering the PIN according to different levels of secure authentication, defined as T-FA (two-factor Authentication) (http://en.wikipedia.org/wiki/Two-factor_authentication). Here, the system authenticates via not only “Something you know” but also “Something you have”. The short of either one disables unauthorized users to access the service due to the entrance control.

In addition, the list of the previously used entrances can also be included and updated in the portable key file. The users will be advised not copying their key files on untrusted machines. However, though the potable keys may be stolen or copied by the others, it yet does not matter, because they cannot steal what you remember like the PIN, as another merit of T-FA. The personal keys will also be periodically updated by the system to avoid possible thefts. This process is easy as the following steps: 1) when a user logs in over a due for key update, the system generates and encrypts the new key file to the user; 2) the

user can use the old key to decrypt and replace the key file with the new one. In case an imposter (suppose being extremely powerful not only stealing the portable key but also knowing your PIN) gets the new key file by pretending as a certain legitimate user, the legitimate user can detect such imposter later on because he cannot access the system using his old key that has been discarded by the system when updating. He can then report to the system administrator for investigation. Thus, the portable key scheme does not increase security risk, but rather reinforces the authentication and security.

3. Modeling and Assessment of the Rebound Wall

3.1 Definitions and Assumptions

It is also important to model, evaluate and analyze the capability of a Rebound Wall against the DoS attacks. We hereby propose an index, Survivability, defined as:
Survivability: the probability that the rebound wall survives under the DoS attacks to continue offering services to legitimate users.

The survivability model is built for measuring the Rebound Wall with assumptions:
Model Assumptions:

1) *N* machines form the Rebound Wall while one machine is the entrance. Each machine has *M* available port numbers.
2) The arrivals of DoS attacks are governed by Poisson processes with a hazard rate λ_{DoS} from external hackers. The external hackers do not know exact IP and Port of the dynamic entrance without Keys, so they only try an objective to launch attacks.
3) Only the entrance machine forwards the requests to the core server, so the DoS attacks cannot make the entrance machine down unless they access to the correct machine on the correct port. Even though the entrance is blocked, the core server is yet fine without directly experiencing the DoS attacks.
4) The other machines of the Rebound Wall monitor the entrance machines by periodically sending pseudo requests to check them. The monitor rate is γ_c.
5) If the entrance is detected down by a pseudo request, the core server will get the report and designate another machine to be the entrance immediately. In the meantime, the system will try to recover the unavailable machine (such as clear buffer, reboot). Suppose the recovery rate is μ_r for each malfunctioning machine.

3.2. Modeling and Evaluation

The above process can be modeled by CTMC (Continuous Time Markov Chain) for the survivability given a *N*-machine rebound wall as Fig. 2.

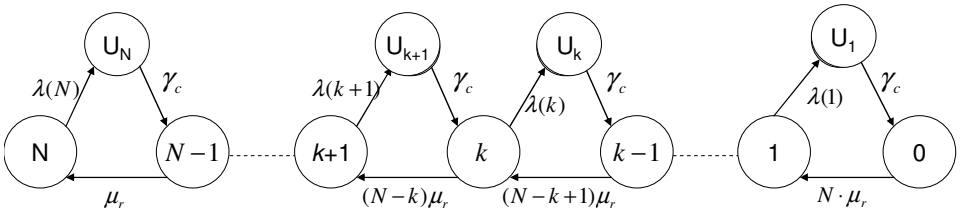

Figure 2: CTMC for the rebound wall with *N* machines.

In this model, State k means there are k rebound wall machines staying at good state, while N-k machines down due to the DoS attacks on a correct entrance (k=0,1,2,…N). State U_i (i=1,2,…,N) are unavailable states after DoS attacks succeed and before the failed entrance is switched to another good entrance. The rate of $\lambda(k)$ can be calculated by

$$\lambda(k) = \frac{1}{kM}\lambda_{DoS} \tag{4}$$

where $\frac{1}{kM}$ is the probability for one DoS attack to accidentally reach the correct entrance given k available machines (k IPs) and M available ports at each machine. According to this model as Fig. 2, the states for the system being unable to offer services to legitimate users include states U_i ($i = 1,2,...,N$) and state 0. Thus,

$$Survivability = \sum_{k=1}^{N} P(k) \tag{5}$$

where $P(k)$ is the probability for the system to stay at the state k. According to the Fig. 2, the *Chapman-Kolmogro*v equations are obtained as

$$\begin{aligned}
&P(N)\cdot\lambda(N) = P(N-1)\cdot\mu_r \\
&P(k)\cdot[\lambda(k)+(N-k)\mu_r] = P(k-1)\cdot(N-k+1)\cdot\mu_r + P(U_{k+1})\cdot\gamma_c \qquad (k=1,2\ldots,N\text{-}1) \\
&P(0)\cdot N\cdot\mu_r = P(U_k)\cdot\gamma_c \\
&P(U_k)\cdot\gamma_c = P(k)\cdot\lambda(k) \qquad (k=1,2\ldots,N) \\
&\sum_{k=0}^{N} P(k) + \sum_{i=1}^{N} P(U_i) = 1
\end{aligned} \tag{6}$$

The *Chapman-Kolmogrov* equations (6) are solvable as follows.

$P(U_k) = P(k)\cdot\lambda(k)/\gamma_c$, so

$P(k) = P(0)\dfrac{N(N-k+1)\mu_r{}^2}{\gamma_c\lambda(k)}$, then let $z_k = \dfrac{N(N-k+1)\mu_r{}^2}{\gamma_c\lambda(k)}$

Thus, $\sum_{k=0}^{N} P(k) + \sum_{k=1}^{N} P(U_k) = P(0) + P(0)\sum_{k=1}^{N}(1+\frac{\lambda(k)}{\gamma_c})\cdot z_k = 1$, Solve it to obtain

$$P(0) = \frac{1}{1+\sum_{k=1}^{N}(1+\frac{\lambda(k)}{\gamma_c})\cdot z_k} \tag{7}$$

and $$P(k) = \frac{z_k}{1+\sum_{k=1}^{N}(1+\frac{\lambda(k)}{\gamma_c})\cdot z_k} \quad (k=1,2,\ldots N) \tag{8}$$

After obtaining all the $P(k)$ s, $\sum_{k=1}^{N} P(k)$ is the survivability as Eq. (5). A numerical example will be illustrated in the later subsection 4.3.

4. Implementation and Case Studies

4.1 Implementation of the Rebound Wall

Different numbers of machines are recruited to form the rebound wall in the following experiments. There is a core server to provide a simple test service that returns the current system time when receiving a request from a legitimate user. Those rebound wall machines are running a very tiny program in background to do simple jobs of switching the requests from the users to the core server, and sending the users the encrypted message about the entrance information. It is the core server that generates the encrypted entrance message and distributes to all rebound wall machines, so the rebound wall machines do not need to do complicated tasks like encryption but only send the stored encrypted entrance message to the users for the first authentication at the user-end (as section 2.3). Therefore, the rebound wall machines consume negligible CPU/Memory resources that do not affect other processes on the host machines. This means that machines forming the rebound wall are not required to dedicate themselves to this protection job. They can still do their own work. In addition, the grid computing technologies (Foster & Kesselman, 2003) can be recruited to detect available machines in the LAN and call them to form the rebound wall.

The period for switching the rebound wall entrance is 10 minutes (600 seconds). The rebound wall checks the availability of the entrance every 10 seconds by sending a pseudo request to see whether it can get the correct service or not. If pseudo request fails to get the correct response within a time, the entrance is switched to another machine. The mean time for valid users to request a service is set to 100 seconds.

In this case study, we use the TFN2K (Tribe Flood Network 2000) packages to launch the DoS/DDoS attacks. TFN2K is a DDoS attack tool that was developed by packet-storm-security (PSS) organization, downloadable from the following link (http://packetstormsecurity.org/groups/mixter/tfn2k.tgz). It consists of two components: master client and daemon. The daemon program is installed on infected computers secretly. The master client program can control multiple daemon agents to launch DDoS attacks toward a specified target by sending commands to daemons secretly from a remote computer. The daemons launch DoS attacks by flooding target with a lot of packets. TFN2k can run on the UNIX, Solaris, and Windows NT platforms.

In order to show the capability and effectiveness of the rebound wall, we made very intensive attacks by TFN2k, *i.e.,* initiated a new DoS attack every 100 seconds to block the valid users' requests on a randomly selected machine of the rebound wall. Please note that in reality, it is impossible for hackers to generate so intensive DoS attacks alternately on the rebound wall machines because they do not have the list of IP addresses that currently form the rebound wall, especially when the grid computing is used to dynamically form the rebound wall with available machines in the LAN.

After the rebound wall is built, some experiments for different case studies were conducted as follows.

4.2 Experiments and Results

Since the purpose of DoS is to block requests from legitimate users, we therefore define the block rate as the percentage of legitimate requests blocked by the DoS attacks, *i.e.,*

$$p_{block} = \frac{\text{The number of requests from legitimate users blocked by the DoS}}{\text{The total number of requests sent from the legitimate users}}$$

We first did an experiment on the system under the DoS attacks without the Rebound Wall. We did 20 rounds and each round had 100 requests sent from legitimate users under the presence of the TFN2K's attacks. The statistics of block rate is given in Table 1.

Table 1: Block rate with and without rebound wall

	Min p_{block}	Max p_{block}	Avg p_{block}	StdDev
No Rebound Wall	62%	83%	74.45%	0.0571
8-machine Wall	0%	5%	0.6%	0.0124

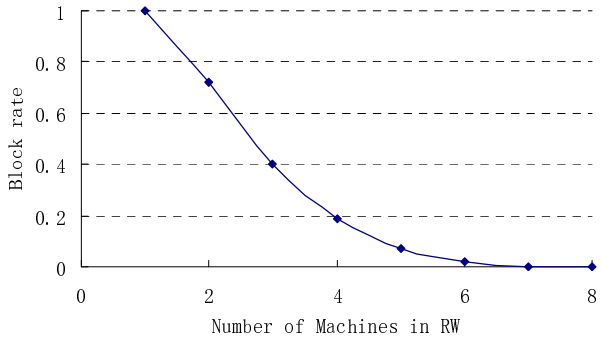

Figure 3: The block rate *vs.* the number of machines in rebound wall (RW)

Then, we did experiments in the presence of an 8-machine rebound wall. The same index of block rate from 20 rounds was shown in Table 1. It is obvious that the rebound wall significantly improved the capability to defend against the DoS attacks. With the 8-machine rebound wall, the average of the block rate is only 0.6%. It means such DoS attacks though intensive are yet vain to the rebound wall, as was almost unable to block legitimate users. Note that the blocking to legitimate users without the rebound wall is caused simply by resources (*e.g.*, buffer) depleted in the server, while with the rebound wall, the blocking is caused by the gap after the entrance is jammed and before it is switched to another machine (as state *U* in Fig. 2).

Then, we analyzed the sensitivity of the parameters to affect the capability of the rebound wall. First, we varied the number of machines in the rebound wall to see the effect on the capability against DoS. The experimental results are depicted in Fig. 3.

We can observe that the number of machines forming the rebound wall affects the block rate. Generally speaking, the more, the better. Nevertheless, we also found that when the number of machines exceeds certain value (here 6 machines), the DoS attacks become useless and the block rate is saturated around 0. This observation exhibits another merit of the rebound wall, *i.e.,* a small rebound wall with only some machines are effective enough to defend against intensive DoS attacks without blocking legitimate users.

Following that, we let the time interval between two checks on the entrance availability vary from 20 seconds to 600 seconds while the other parameters remained unchanged as above. The experimental results are plotted in Fig. 4. From the results, we find that less monitoring frequency causes a higher block rate.

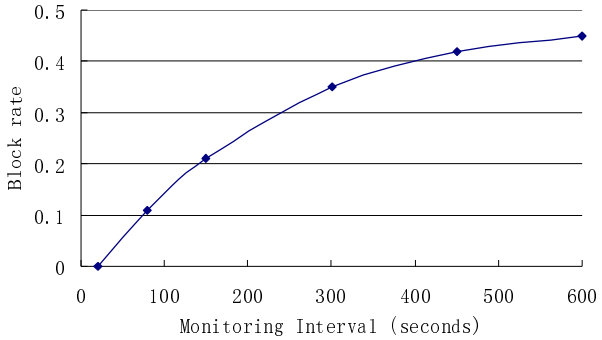

Figure 4: Block rate *vs.* monitoring intervals.

4.3 Survivability Analysis

From Table 1, it was observed that the results of the block rate were different in all the experiments with about (0.01 to 0.06) standard deviation. Also, from the Fig. 3, we found that the block rate was saturated at early points to reach 0%. In fact, it does not mean the rebound wall is perfect, but means that no users' requests arrived when the rebound wall was being temporarily undermined or in switch mode. Therefore, the experimental results only gave us intuitive feeling on the effectiveness and ability of the Rebound Wall, but cannot theoretically prove or validate the capability due to the randomness and saturation.

Therefore, the analytical results of survivability are derived as a theoretical index for measuring the capability of the rebound wall according to subsection 3.2. We also illustrate how to collect the parameters needed by the survivability model in practice.

In this case study, the DoS attack has a mean arrival time of 100 seconds, so $\lambda_{DoS} = 0.01\ s^{-1}$. The mean cycle time for monitoring and switching rate here is 10 seconds, so $\gamma_c = 0.1\ s^{-1}$. If any machine is down due to the DoS attack, we repair it by the following steps: (1) the DNS disconnects it from the Internet to make the DoS attacks lose target, (2) in the meantime, clean the buffer, (3) reconnect it. If possible (such as dynamical IP by DNS), its IP is updated. The experiments showed the average time for such repair is 40 seconds by each machine, so $\mu_r = 0.025 s^{-1}$.

These parameters are entered into the Markov model as Fig. 1, we can calculate by Eq. (5) and Eq. (8) the survivability as 0.9693 for the 8-machine rebound wall.

The survivability to different numbers of rebound wall machines (*N* from 4 to 14) and to various monitoring and switching rate $\gamma_c \in [0.01,0.2]$ is depicted by Fig. 5.

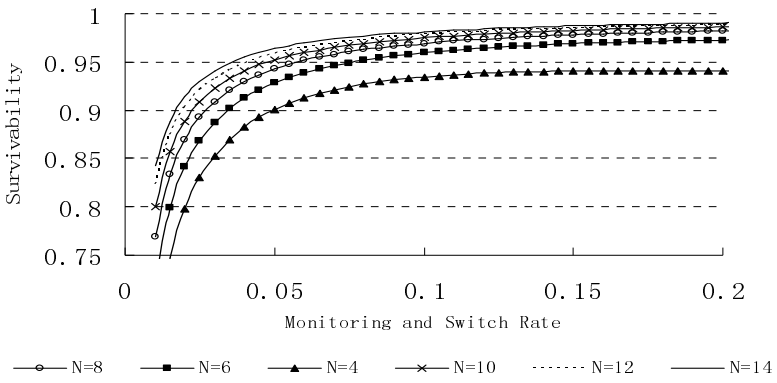

Figure 5: Survivability *vs.* γ_c given different number of RW machines (*N*)

5. Conclusion and Discussions

DoS/DDoS attack is one of the most critical security problems in today's network systems, which is easy to launch by hackers but hard to defend by victims. This paper is the first to present a novel mechanism of Rebound Wall which proved very effective to protect a core server from DoS/DDoS attacks and easy to implement/deploy in practice. The core server does not directly receive the requests from the external users, but is protected by a rebound wall formed by available machines in the LAN. Those designated entrance machine(s) receive(s) valid requests from authorized users and transfer(s) the requests to the server. These entrance machines are roaming over the rebound wall. As a result, illegitimate hackers lose target for attack while legitimate users can get the correct IP and Port of the activated entrance machines according to the new presented mechanisms: user-end authentication by key and entrance-based privilege control. The monitoring function by Pseudo requests can efficiently detect problems and entrance switch can immediately recover the service to authorized users in case the entrance is violated or failed. The portable key scheme not only increases the flexibility and portability for users, but also enhances the security levels for authentication according to T-FA. Moreover, the rebound-wall technology endowed strong capability for intrusion detection by simply checking wrong knocks on the rebound wall. It can also help trigger the trace back.

Acknowledgement: Authors would like to thank the anonymous referees who helped improve the paper. This work was supported in part by NSF under Grant # 0831609.

References

[1] Amir, Y., Nita-Rotaru, C., Stanton, and S., Tsudik, G., *Secure spread: an integrated architecture for secure group communication*, IEEE Transactions on Dependable and Secure Computing, Vol. 2, No. 3, pp. 248 – 261, 2005.

[2] Badishi, G., Keidar, I., and Sasson, A., *Exposing and eliminating vulnerabilities to denial of service attacks in secure gossip-based multicast*, IEEE Transactions on Dependable and Secure Computing, Vol. 3, No. 1, pp. 45 – 61, 2006.

[3] Bencsath, B. and Vajda, I., *Protection against DDoS attacks based on traffic level measurements*, Western Simulation MultiConference, pp. 22-28, 2004.

[4] Bishop, M., *Computer Security: Art and Science*, Second Edition, Addison Wesley, ISBN 0-201-44099-7, 2003.
[5] Dai, Y.S., Xie, M., and Poh, K.L., *Modeling and analysis of correlated software failures of multiple types*, IEEE Transactions on Reliability, Vol. 54, No. 1, pp. 100-106, 2005.
[6] Foster, I. and Kesselman, C., *The Grid 2: Blueprint for a New Computing Infrastructure*, Morgan-Kaufmann, 2003.
[7] Ioannidis, J. and Bellovin, S.M., *Implementing pushback: Router-based defense against DDoS attacks*, in NDSS Conference Proceedings, pp. 6-8, 2002.
[8] Jin, G., Wang, H., Shin, K. G., *Hop-count filtering: an effective defense against spoofed DDoS traffic*, In Proceedings of the 10th ACM conference on Computer and communication security (CCS), pp. 30-41, 2003.
[9] Keromytis, A.D., Misra, V., and Rubenstein, D., *SOS: an architecture for mitigating DDoS attacks*, IEEE Journal of Selected Areas in Communications, Vol. 22, No. 1, pp. 176-188, 2004.
[10] Khattab, S.M., Sangpachatanaruk, C., Mosse, D., Melhem, and R., Znati, T., *Roaming honeypots for mitigating service-level denial-of-service attacks*, in Proceedings of ICDCS'04, pp. 328-337, 2004.
[11] Krawetz, N., *Anti-honeypot technology*, Security & Privacy Magazine, IEEE, Vol. 2, No. 1, pp. 76- 79, 2004.
[12] Li, Q., Zhu, H., Zhang, M., and Ju, J., *Simulating and improving probabilistic packet marking schemes using Ns2*, IEEE Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05), pp. 348-352, 2005.
[13] Moore, D., Shannon, C., Brown, D. J., Voelker, G. M, and Savage, S., *Inferring Internet denial-of-service activity*, ACM Transactions on Computer Systems, Vol. 24, No. 2, pp. 115–139, 2006.
[14] Perrig, A., Song, D., and Yaar, A., *StackPi: a new defense mechanism against IP spoofing and DDoS attacks*, CMU Technical Report, 2002.
[15] Mirkovic, J., Robinson, M., Reiher, P., and Kuenning G., *Alliance Formation for DDoS Defense*, Proceedings of the New Security Paradigms Workshop, ACM SIGSAC, pp. 11-18, August 2003.
[16] Spitzner, L., *The honeynet project: trapping the hackers*, Security & Privacy Magazine, IEEE, Vol. 1, No. 2, pp. 15- 23, 2003.
[17] Wang, H., Zhang, D., and Shin, K. G., *Detecting SYN flooding attacks*, in Proceedings of IEEE INFOCOM, Vol. 3, pp. 1530-1539, June 23-27, 2002.
[18] Xiang, Y. and Zhou, W., *Defense system against DDoS attacks by large-scale IP traceback*, IEEE Third International Conference on Information Technology and Applications (ICITA'05), Vol. 2, pp. 431-436, 2005.
[19] Xie, M., Dai, Y.S., and Poh, K.L., *Computing Systems Reliability: Models and Analysis*, Kluwer Academic Publishers: New York, NY, U.S.A., 2004.
[20] Zhang, S., Dasgupta, P., *Denying denial-of-service attacks: a router based solution*, International Conference on Internet Computing, pp. 301-307, 2003.
[21] Zou, X., Ramamurthy, B., Magliveras, S., *Secure Group Communication over Data Networks*, Springer, 2004.

Yuanshun Dai received the B.S. degree from the Automation Department at Tsinghua University in 2000 and the Ph.D. degree from the National University of Singapore, in 2004. He is currently an assistant professor with the Department of Industrial and Information Engineering and the Department of Electrical Engineering and Computer Science at the University of Tennessee, Knoxville. Before that, he was an assistant professor in the Department of Computer and Information Science at the Purdue University School of Science, Indiana University, Purdue University, Indianapolis, U.S.A. He was the program chair of the 12th IEEE Pacific Rim Symposium on Dependable

Computing (PRDC 06). He was also the founder and general chair of the 1st, 2nd, and 3rd IEEE Symposium on Dependable Autonomic and Secure Computing (DASC 2005, 2006, 2007). He is a guest editor of the IEEE Transactions on Reliability and is on the editorial board of some other international journals. His research interests are dependability, security, grid computing, and autonomic computing. He is the author or coauthor of over 60 papers and 4 books.

Xueping Li is an Assistant Professor of the Department of Industrial and Information Engineering and the Director of the Intelligent Information Engineering Systems Laboratory (IIESL) at the University of Tennessee - Knoxville. His research areas include information assurance, scheduling, and supply chain management. He received the B.S. degree and the M.S. degree from the Computer Science Department at Nankai University. He received the Ph.D. degree from the Department of Industrial Engineering at Arizona State University. He is a member of IEEE, IIE and INFORMS.

Xukai Zou is a faculty member with the Computer Science Department of Purdue University School of Science, at IUPUI, U.S.A. He received his Ph.D. from University of Nebraska-Lincoln and M.S. and B.S. from Huazhong University of Science and Technology and Zhengzhou University respectively, all in Computer Science. His research is in Applied Cryptography, Communication Networks and Security, and Grid Computing. He has published three books and over twenty articles in these areas. Dr. Zou is a recipient of the U.S. NSF Cyber Trust Awards and the leading author of the Book "Secure Group Communication over Data Networks" (Springer). Dr. Zou is a Program Chair and a PC Member for a number of international conferences and serves on the Editorial Board and as a reviewer for many international journals and conferences. He is a member of IEEE.

Liudong Xing is a faculty member with the Electrical and Computer Engineering Department, University of Massachusetts, Dartmouth. She received her M.S. and Ph.D. degrees from the University of Virginia (U.Va.) in 2000 and 2002, respectively. Dr. Xing served as a program co-chair for IEEE DASC'06, a program vice chair for ICESS'07 and ICPADS'08, and an associate guest editor for the Journal of Computer Science in 2006. She is the Editor for Short Communications in the International Journal of Performability Engineering. Dr. Xing is the recipient of the IEEE Region 1 Technological Innovation (Academic) Award in 2007. Her research interests include dependable computing and networking, reliability engineering, and sensor networks. She is a senior member of IEEE.