A Distributed Framework for Defending Denial of Service Attacks

Yen-Hung Hu
Department of Computer Science
Hampton University
Hampton, Virginia 23668
Email: yenhung.hu@hamptonu.edu

Abstract—This paper investigates the behavior of DoS attacks while they are affecting the stability of computer systems as well as exploit a distributed framework which will monitor, detect, and prevent DoS attacks. This framework includes several distributed monitors implemented at the critical components and a management center which is used to detect and prevent DoS attacks.

I. INTRODUCTION

Denial of Service (DoS) attack refers to any techniques that prevent a resource from being used by legitimate users. Such a resource may be located at a host domain or a network domain. It could be Central Processing Unit (CPU) computation power, memory, communication bandwidth, communication port, etc. The malicious activities (or even some legitimate activities) can trigger plenty of faulty service requests to exhaust most of the resources, as a result services of legitimate requests are denied.

DoS attacks can cause computers, routers, service providers, and even the entire network to experience serious instability and performance degradation. Some of the attacks exploit system vulnerabilities [1], [2], [3], [4], [5], some of them impose computationally intensive requests on victims [1], [2], [3], [4], [5], [6], [7], and some of them use flooding-based (distributed) DoS attacked [1], [2], [3]. Typically, attacks induced by system vulnerabilities could be easily fixed by updating with the newest patches of hardware and software released for such issues. Issues caused by computation intensive tasks could be reduced by deploying countermeasures such as cookie in the Internet Key Exchange and anti-replay protection in IPSec [8], [9]. Influence of flooding-based DoS attack also could be mitigated if routers and every individual computer have the capacity to monitor and filter every transactions and packets.

In this paper, we implement the monitors at critical components. We define a critical component as a part of a program, a system, or a network, whose performance will be severely downgraded when this critical component is unable to perform its regular operations. For example, CPU is a critical component of a computer. In case this CPU reaches its maximum utilization, long delay will occur to all of its waiting operations.

A monitor will record four entities of a critical component when a new request arrives. Such entities are request ID, request time, service size, and utilization. Request ID is to identify which process makes this request. Request time is to represent when this request arrives. Service size is to indicate job size of this request. Utilization is to show usage of the critical component.

Our framework will integrate information collected from the distributed monitors to identify and prevent DoS attacks. The ideas of our detection models and management algorithms will be included.

The rest of this paper is organized as follows. In section II, we study several related works. In section III, a distributed framework used to detect and defend DoS attacks is demonstrated. Section V concludes the paper.

II. RELATED WORK

Liedtke et al. [10] have given a different view of studying DoS attacks. They have examined a specific type of malicious activity: DoS attacks using legal system operations. They have evaluated their algorithms upon L4 kernel with direct attacks and indirect attacks. In case resources are maintained and allocated by servers, they proposed two ways to defend DoS attacks: (a) all resources can be managed by servers; and (b) servers can be constructed in such a way that they can not be blocked by DoS attacks. Therefore, servers must be able to protect resources against any improper abuse. However, since this study concentrates on a specific type of DoS attacks and on a specific hardware and operating system, it would not cover all issues relating to DoS attacks.

Cabrera et al. [11] have developed a framework for detecting flooding-based distributed DoS attacks. They have investigated a three-step-mechanism: detecting attacks; detecting correlations; and detecting precursors to attacks. Such a mechanism could be useful for varies detection systems if the detection algorithms are properly implemented. They have adopted Granger Causality Test [12] to exam the residual of the autoregressive model corresponding to the key variables at the target and the residual of the moving average model corresponding to the input-output pairs, where the output is the key variable at the target and the input is one of the Management Information Base (MIB) variables at the attackers. Because, in their project, identifying key variables of the attackers and targets may not be feasible when attackers are unknown, it could still be improved.
III. DISTRIBUTED FRAMEWORK

In this paper, we depict a distributed framework (see figure 1) to detect and defend DoS attack. This framework includes appropriate amount of monitors and a management center. The monitors are implemented at critical components to record information of every event and will send such an information to the management center during every constant period. The management center will on-line/off-line process the information collected from the monitors and determine the status of each critical component and each running process. When the management center detect the suspicious processes, it will inform the monitors which reside in the infected critical components to disrupt the services provided for such suspicious processes.

A. Algorithms

Our distributed framework includes four phases: monitor (see algorithm 1), management (see algorithm 2), detecting DoS attack (see algorithm 3), and identifying malicious process (see algorithm 4). Algorithms of them are as follows.

Algorithm 1 MONITOR

1: \( \exists \) a monitor \( m_i \) at critical component \( i \)
2: For every \( \delta \tau_j \) seconds, a \( B_i = \{b_1, b_2, \ldots, b_k\} \) is sent to the management center. \( B_i \) could be the set of request \( ID \), request time, service size, utilization, etc.
3: for all process \( j \) in critical component \( i \) do
4: if \( j \in Z_t \) then
5: disrupt the service provided for process \( j \)
6: end if
7: end for

Algorithm 2 MANAGEMENT CENTER

1: for all \( B_i \) collected from the monitor \( i \) do
2: for all process \( k \in B_i \) do
3: conduct these two algorithms: "DETECTING DOS ATTACK" AND "IDENTIFYING MALICIOUS PROCESS"
4: end for
5: end for
6: if process \( k \) is a suspicious DoS attack then
7: Add \( k \) into the suspicious set \( Z_t \)
8: Update \( Z_t \) to all monitors
9: end if

Algorithm 3 DETECTING DOS ATTACK

1: \( \exists \) a DoS attack \( x \)
2: if \( \varphi_x > \pi_x \) then
3: The system is attacked by the DoS attack \( x \)
4: end if
5: where, \( \varphi_x = \sum_{i=1}^{m} \sum_{j=1}^{m} p_x^i \sum_{k=1}^{n} p^{kj} \cdot q^{ix} \) denotes the probability of \( j^{th} \) critical component which is also saturated when \( i^{th} \) critical component is saturated. \( q^{ix} \) denotes the weight of critical component \( i \) while the system is affected by DoS attack \( x \).

B. Monitor

Let \( m_i \) be the monitor residing in the critical component \( i \), and \( c_i \) is the control implemented inside \( m_i \). \( c_i = \{t, IDS\} \) means that after time \( t \) the request from any process whose \( ID \) is in the set, \( IDS \), will be denied. If \( c_i = \{\} \), all requests to the critical component \( i \) are granted.

\( B_i = \{b_1, b_2, \ldots, b_k\} \) denotes a table created by \( m_i \), where \( b_j \) is the \( j^{th} \) critical component \( i \), entity used to record the service request information. Such information includes request \( ID \), request time, service size, and utilization. For every \( \delta \tau_i \) seconds, message \( B_i \) of the critical component \( i \) will be encrypted and sent to the management center.

To simplify these issues, we consider that processes (programs) are the only candidate of launching DoS attacks, no matter whether these attacks are inside or outside the system. However, request \( ID \) in the table \( B_i \) will be different. When the attack is inside the system, its request \( ID \) should relate

Algorithm 4 IDENTIFYING MALICIOUS PROCESS

1: while DoS Attack \( x \) is detected do
2: for all process \( j \) do
3: if \( G^j > \text{threshold} \) then
4: add \( j \) into \( Z_t \)
5: end if
6: end for
7: end while
8: where, \( G^i = \sum_{i=1}^{m} g^i_j \cdot q^j_t = \frac{(R_0 - R_1) / \tau}{R_1 (T - 2T - 1)} \cdot R_0, R_1, T, \) and \( \tau \) are calculated by using GCT.
to the unique process ID assigned by the operating system. When the attack is outside the system, its request ID should be able to reveal the information of IP address, protocol number, port number, and application used. To solve these issues, the monitors should be able to identify internal attacks and external attacks. They also should be able to communicate with the management center with low cost (low system delay and low system performance degradation).

C. Management Center

In the paper, the management center will perform two processes: off-line batch process and on-line real time process. The off-line batch process is used to analyze the information collected from the monitors and to create the models for identifying DoS attacks. The on-line real time process is used to examine new arriving messages and to update the control of each monitor. For example, if a process k has been identified as a DoS attack by the off-line batch process, the management center will update the controls at every monitor. Therefore, all monitors will block the requests from process k.

Let \( Z_t \) denote a set of suspicious DoS attacks identified after time \( t \). If process \( z \in Z_t \), then, after time \( t + \delta t \), all requests from the process \( z \) will be blocked by the monitor \( m_i \), for all \( i \). Where, \( \delta t \) is the delay of updating controls.

D. Detecting DoS Attack

At first, we develop a linear matrix model to detect DoS attacks. Assume there are \( m \) critical components in the system. For each component, there are \( n \) factors that could be used to represent the attacking scenarios of DoS attack \( x \). We assume that the system is affected by DoS attack \( x \) if the following conditions are satisfied:

\[
W^x = P_1^x Q^x + P_2^x Q^x + ... + P_n^x Q^x = Q^x \sum_{k=1}^{n} P_k^x
\]

\[
\varphi_x = \sum_{i=1}^{m} w^x_i > \pi_x
\]

where, \( W \) is a \( 1 \times m \) matrix, \( P \) is a \( m \times m \) matrix, and \( Q \) is a \( 1 \times m \) matrix. \( P_k^x \) denotes \( k \)th factor matrix that is distinguished while the system is affected by DoS attack \( x \). It can be rewritten as: \( P_k^x | p_{ij}^{kx} | \leq 1, i \in (1, m), j \in (1, m), k \in (1, n) \), where, \( p_{ij}^{kx} \) denotes the probability of \( j \)th critical component which is also saturated when \( i \)th critical component is saturated (i.e., value is larger than the specific threshold). \( Q^x | q_i^x | \leq 1, i \in (1, m) \) denotes the weight of each critical component while the system is affected by DoS attack \( x \).

\[
W^x | w^x_i, i \in (1, m) \) denotes the contribution of each component to DoS attack \( x \). In equation (2), \( \varphi_x, \varphi_x \leq m \), denotes the probability that the system is affected by DoS attack \( x \). \( \pi_x, x \leq m \), represents the threshold (The project will study and investigate the threshold of each DoS attack). Therefore, \( \varphi_x > \pi_x \) means the system is affected by DoS attack \( x \).

For example, assume that utilization (\( P_1^x \)) and service size (\( P_2^x \) are the only two factor matrices (n=2) that are distinguished when the system is affected by DoS attack \( x \). Therefore, equation (1) can be simplified as \( W^x = P_1^x \cdot Q^x + P_2^x \cdot Q^x \) and can be represented as

\[
W^x = \begin{bmatrix}
  w_1^x \\
  \vdots \\
  w_m^x
\end{bmatrix} = \begin{bmatrix}
  p_{11}^{x} & \cdots & p_{1m}^{x} \\
  \vdots & \ddots & \vdots \\
  p_{m1}^{x} & \cdots & p_{mm}^{x}
\end{bmatrix} \begin{bmatrix}
  q_1^x \\
  \vdots \\
  q_m^x
\end{bmatrix} + 
\begin{bmatrix}
  \sum_{i=1}^{n} p_i^{x} \\
  \vdots \\
  \sum_{i=1}^{n} p_n^{x}
\end{bmatrix} \begin{bmatrix}
  q_1^x \\
  \vdots \\
  q_m^x
\end{bmatrix}
\]

Hence, \( w^x_i \) can be written as:

\[
w_1^x = (p_{11}^{x} + p_{12}^{x}) q_1^x + (p_{11}^{x} + p_{12}^{x}) q_2^x + \cdots + (p_{11}^{x} + p_{12}^{x}) q_m^x
\]

\[
w_2^x = (p_{21}^{x} + p_{22}^{x}) q_1^x + (p_{21}^{x} + p_{22}^{x}) q_2^x + \cdots + (p_{21}^{x} + p_{22}^{x}) q_m^x
\]

\[
\vdots
\]

\[
w_m^x = (p_{m1}^{x} + p_{m2}^{x}) q_1^x + (p_{m1}^{x} + p_{m2}^{x}) q_2^x + \cdots + (p_{m1}^{x} + p_{m2}^{x}) q_m^x
\]

When there are \( n \) factors of each critical components could be used to represent DoS attack \( x \). We get:

\[
w_1^x = (p_{11}^{x} + p_{12}^{x} + \cdots + p_{1n}^{x}) q_1^x + (p_{11}^{x} + p_{12}^{x} + \cdots + p_{1n}^{x}) q_2^x + \cdots + (p_{11}^{x} + p_{12}^{x} + \cdots + p_{1n}^{x}) q_m^x
\]

\[
w_2^x = (p_{21}^{x} + p_{22}^{x} + \cdots + p_{2n}^{x}) q_1^x + (p_{21}^{x} + p_{22}^{x} + \cdots + p_{2n}^{x}) q_2^x + \cdots + (p_{21}^{x} + p_{22}^{x} + \cdots + p_{2n}^{x}) q_m^x
\]

\[
\vdots
\]

\[
w_m^x = (p_{m1}^{x} + p_{m2}^{x} + \cdots + p_{mn}^{x}) q_1^x + (p_{m1}^{x} + p_{m2}^{x} + \cdots + p_{mn}^{x}) q_2^x + \cdots + (p_{m1}^{x} + p_{m2}^{x} + \cdots + p_{mn}^{x}) q_m^x
\]

In general, \( w^x_i \) can be rewritten as:

\[
w_i^x = \sum_{j=1}^{m} q_i^x \sum_{k=1}^{n} p_{jk}^{x}, \forall i \in (1, m)
\]

Therefore, for DoS attack \( x \), we get

\[
\varphi_x = \sum_{i=1}^{m} w_i^x = \sum_{i=1}^{m} q_i^x \sum_{k=1}^{n} p_{jk}^{x}
\]

If \( \varphi_x > \pi_x \), then the system is affected by DoS attack \( x \). Since different DoS attacks will have different attack patterns, therefore \( Q \) need to be adjusted dynamically. Intuitively, this model can be easily applied to a system with multiple DoS attacks.
E. Identifying Malicious Process

When DoS attack \( x \) is detected, we now need to identify which processes cause this attack. In this paper, we adopt the Granger Causality Test (GCT) \([11, 12]\) to investigate this problem.

GCT is used to test whether lagged information on a variable \( u \) provides any statistically significant information about a variable \( v \). In this paper, we attempt to determine which process IDs trigger the DoS attack \( x \). There may be multiple DoS attacks in the system. In the first step, we need to determine the variables at the attacker (malicious process) that are causally related to DoS attack processes when the system is affected by DoS attack.

To simplify the problem, in here, we only consider one DoS attack (i.e., DoS attack \( x \)) at a time. Let's assume that the system is affected by only one DoS attack. Let \( U_i \) denote the key variable (e.g., utilization) of critical component \( i \), and \( R^j_i \) denote the key variable (e.g., service size) of process \( j \) at the critical component \( i \). When the system is affected by DoS attack \( x \), and assume a particular lag length \( \tau \), for process \( j \) and critical component \( i \), we have the GCT equation

\[
U_i(k+1) = \sum_{l=1}^{\tau} \alpha_l U_i(k-l+1) + \sum_{l=1}^{\tau} \beta_l R^j_i(k-l+1) + e(k) \tag{5}
\]

The Null Hypothesis is given when \( P_l = 0, l = 1, 2, ..., T \). Therefore, the Null Hypothesis of the GCT equation is

\[
U_i(k + 1) = \sum_{l=1}^{\tau} \gamma_l U_i(k - l + 1) + \delta(k) \tag{6}
\]

Let \( R_1 = \sum_{t=1}^{T} \epsilon^2(t) \) and \( R_0 = T \sum_{t=1}^{T} \epsilon^2(t) \) then the test statistic is given by:

\[
g^j_i = \left( \frac{R_0 - R_1}{T}\right) / \left( \frac{R_1}{T - 2\tau - 1}\right) \approx F(\tau, T - 2\tau - 1) \tag{7}
\]

where, \( g^j_i \) denotes the test statistic of process \( j \) at critical component \( i \). \( F(\tau, T - 2\tau - 1) \) is the Fisher's \( F \) distribution \([12]\) with parameter \( \tau \) and \( T - 2\tau - 1 \).

Now, we consider the causality of process \( j \) at all critical components when the system is affected by DoS attack \( x \). We get

\[
G^j = \sum_{i=1}^{m} g^j_i \tag{8}
\]

If \( G_j \) is greater than the critical value, then we say that process \( j \) is causally related to the high key variable (i.e., utilization) of all critical components. It means that process \( j \) could be causally related to DoS attack \( x \). If more than one process has test statistic value greater then the critical value, all of them are causally related to DoS attack \( x \). However, if \( G^i > G^j \), we say that process \( i \) is more likely than process \( j \) to be causally related with DoS attack \( x \). Intuitively, this model could easily be applied to the case that the system is affected by multiple DoS attacks.

IV. Conclusion

In this paper, we propose a distributed framework which could be used to detect and prevent DoS attacks. This framework includes four phases: monitor, management center, detecting DoS attack, and identifying malicious process. We have depicted the algorithms of these four phases and included the rationales to support our projections.

References