Vulnerable Path Attack and its Detection

Chuyu She[1, 2, 3], Wushao Wen[1, 2], Quanqi Ye[1] and Kesong Zheng[1]
[1] School of Data and Computer Science, Sun Yat-sen University,
Guangzhou 510006, PR China
[e-mail: shechuyu@gdufe.edu.cn]
[2] SYSU-CMU Shunde International Joint Research Institute,
Shunde 528300, PR China
[e-mail: wenwsh@mail.sysu.edu.cn]
[3] School of Mathematics and Statistics, Guangdong University of Finance & Economics,
Guangzhou 510320, PR China
*Corresponding author: Wushao Wen

Received July 1, 2016; revised January 11, 2017; accepted February 7, 2017;
published April 30, 2017

Abstract

Application-layer Distributed Denial-of-Service (DDoS) attack is one of the leading security problems in the Internet. In recent years, the attack strategies of application-layer DDoS have rapidly developed. This paper introduces a new attack strategy named Path Vulnerabilities-Based (PVB) attack. In this attack strategy, an attacker first analyzes the contents of web pages and subsequently measures the actual response time of each webpage to build a web-resource-weighted-directed graph. The attacker uses a Top M Longest Path algorithm to find M DDoS vulnerable paths that consume considerable resources when sequentially accessing the pages following any of those paths. A detection mechanism for such attack is also proposed and discussed. A finite-state machine is used to model the dynamical processes for the state of the user's session and monitor the PVB attacks. Numerical results based on real-traffic simulations reveal the efficiency of the attack strategy and the detection mechanism.

Keywords: Application-layer DDoS attack, Path vulnerabilities-based attack, Web-resource-weighted-directed graph, Top M longest path algorithm, Finite-state machine

This research was supported by the Science and Technology Project of Guangdong province of PR China (2014B010114002，2015B010108004)

https://doi.org/10.3837/tiis.2017.04.018 **ISSN : 1976-7277**

1. Introduction

Denial-of-Service (DoS) attacks target the vulnerabilities of a system by using special crafted traffic, or deplete the system resources or network bandwidth with overwhelming packets. These kinds of attacks result in denial of services or crashed targeted systems. As computers and network technology evolve, system vulnerabilities are patched and outstanding ones are less prone to exploitation. Moreover, system processing capability improves, memory size increases, and network bandwidth expands. These technological advances increase the difficulty of DoS attacks. Distributed DoS (DDoS) attacks emerge as a result.

A DDoS attack is a distributed and coordinated attack that evolved from DoS. This attack leverages a large number of computers to launch attacks to the victim. A DDoS attack is more effective and difficult to prevent. These kinds of attacks are performed at the network layer or the application layer. Network-layer DDoS attacks, such as ICMP flooding, SYN flooding and UDP flooding [1], mainly exploit the vulnerabilities of protocols of network and transport layer. However, with the advancements of defense technology, DDoS attackers are gradually targeting the application layer. Application-layer DDoS attacks mainly exploit the vulnerabilities in application layer protocols and target the resources of this layer. These kinds of attacks are complicated and difficult to detect because they take advantage of the complexity and diversity of application protocols.

Traditional DDoS defense systems are not effective in protecting application-layer DDoS attacks because most methods against DDoS attacks focus on the transport layer, network layer, or lower layers [2-9]. These protecting methods are divided into two categories: statistical approaches based on network flows and packets, and algorithms based on signatures. Statistical approaches based on network flows and packets are effective in protecting against DDoS attacks at the transport and lower layers. However, they cannot distinguish the flows and packets of an application-layer DDoS attack from those of a normal application because these approaches are incapable of analyzing application layers. Algorithms based on signatures can detect attacks at the application layer. However, they only detect known attacks and are incapable of detecting zero-day attacks. These methods cannot provide sufficient protection against application-layer DDoS attacks.

Application-layer DDoS attacks have two types: attacking resources [10-13] and attacking software vulnerabilities [14-17]. Models are devised for defending against attacks to the resources at the application layer. However, these models are mostly used to protect against random attacks or attacks of repeating requests for specific pages as mentioned by Gavrilis [12]. An attacker who constantly requests a random web page is detected by measuring the variance of each click stream. However, this detection mechanism can be eluded when the attacker scans the website and launches the attack following the web hyperlinks that resemble real human web users.

In this paper, we propose a covert and effective path-based asymmetric traffic attack. This kind of attack better mimics the normal behavior of web users and consumes a large amount of resources with minimal amount of attack traffic. In this attack strategy, an attacker first analyzes the contents of all the web pages of a website and then measures the actual response time of each new webpage from the previous webpage by probing the server response time. A weighted directed graph is used to model the response time and relationship of webpages. Based on our observation and analysis, when a webpage is accessed, a longer response time normally results in additional resource consumption. According to these assumptions and the

weighted directed graph, the attacker can use a Top M Longest Path (TMLP) algorithm to find M vulnerable paths of the website and trigger great resource consumption when the pages are sequentially accessed following one of the chosen paths. We named this new attack method as the Path Vulnerabilities-Based (PVB) attack. A PVB attack is detectable if the mechanism is well understood. Many differences are observed between the sessions of normal users and attackers. An efficient defense mechanism that protects the system from PVB attack is also proposed. Four factors are considered in our detection strategy: repeat times of a request sequence, average response time, average popularity, and average transition probability of the request resource. The finite-state machine (FSM) is used to model the dynamical processes for the state of the user's session and monitor the PVB attacks for the accurate detection of PVB attacks.

The rest of this paper is organized as follows. In Section 2, we summarize the works related to our research. In Section 3, we outline the ways to analyze the contents of a website and present a TMLP algorithm to search for the vulnerable paths of a website. In Section 4, we introduce the PVB attack and the defense model for such an attack. In Section 5, we validate the efficiency of PVB attack using real-traffic simulations. Finally, in Section 6, we conclude our work and explore potential future research topics.

2. Related Work

As discussed, DoS and DDoS attacks are serious problems on the Internet. In recent years, network-security researchers have developed quite a few methods to defend against DDoS attacks. Simmross et al. [2] used statistical models to detect DDoS attacks. Mohammed et al.[3] presented UPPM as a novel probabilistic packet marking scheme for IP traceback, that is suitable for finding the path from a victim back to the attacker. Xiang et al. [4] used information theory and proposed two new information metrics to detect low-rate DDoS attacks. Shiaeles et al. [5] proposed a method for DDoS detection using fuzzy estimators. Kang et al. [6] suggested a DDoS defense system called sShield that works inside an AS using traffic deflection method similar in principle to that of the Shield. Cattani [9] analyzed new progress in detecting the anomalies in network flow and DDoS attacks by using wavelet technologies.

As network layer DDoS defenses become effective, DDoS attackers are gradually targeting the application layer. Ranjan et al. [10] characterized application-layer resource attacks as either request flooding, asymmetric, or repeated one-shot. HTTP flooding attack is a type of application-layer-resource attack. It sends a large number of malicious HTTP requests to a target server and finally exhausts the resources such as network bandwidth [11].

Application-layer attacks are increasing, such as HTTP GET request floods, "mail bombs," or floods from spam networks. DNS-based attacks are also popular, in which attackers flood the DNS servers with bogus but well-formed requests [13]. Recently, application-layer attacks have been engineered to target vulnerabilities in protocols. An example is Slowloris [14], which is based on a weakness in HTTP GET. A more recent attack is the SlowPOST, which targets the vulnerabilities in HTTP POST. McGregory also mentioned some recent attacks, such as Rudy, SSL renegotiation attack, VoIP flood, and IPv6 extension header fragmentation [17].

Detection of application-layer DDoS attacks is well discussed. Nam et al. [18] proposed a new DDoS defense mechanism that protects HTTP web servers from application-level DDoS attacks based on two methodologies: whitelist-based admission control and busy period-based

attack flow detection. Yoon [19] proposed a novel DDoS mitigation scheme for critical Internet sites by building a whitelist. Ranjan et al. [10] proposed a counter-mechanism, DDoS Shield, which consists of a suspicion assignment mechanism and a DDoS-resilient scheduler. Ramamoorthi et al. [20] proposed an anomaly-detection mechanism to detect DDoS attacks by using Enhanced Support Vector Machine with string kernels. However, these methods are not well designed for attacks that follow web links. Therefore, our proposed path-based asymmetric traffic attacks easily bypass those protection methods. We will discuss a new effective protection mechanism from such attacks.

3. Vulnerable Path Analysis of Web Pages

3.1 Webpage weaklink modeling

We obtain all the pages of a website by using web crawlers. We obtain a directed graph of the website by analyzing the contents of the webpages. Probing on the website is triggered to obtain the actual access time of each webpage from a previously visited webpage by following a link from the directed graph and using the access time as the weight for the link. We then construct a weighted directed graph to model the relationships of web pages on the website. The weighted directed graph $G = (V, E, w)$ is built as follows.

3.1.1 Vertices (V)

The web resources are extracted as nodes set V of the weighted directed graph G by analyzing the contents of webpages of the targeted website. The word resource mentioned in this study has a flexible meaning: it can be a new webpage, a picture, a video, a voice, or a text. A unique identification (ID) is assigned to each resource node for identification. To describe a resource node in detail, we use the URI address, the type (C), and the size (S) as the attributes of the resource node. The type can be html, jpg, gif, png, bmp, or pdf. The set of vertices $V = \{v_1, v_2, \cdots, v_n\}$ is used to represent the web resources. Each element of V corresponds to a tuple (ID , URI , C , S).

3.1.2 Edges (E)

Edge E of the weighted directed graph G is obtained by analyzing the link relationship between resource nodes. If the resource v_i can link to the resource v_j, then $e(v_i, v_j) \in E$. Otherwise, $e(v_i, v_j) \notin E$.

3.1.3 Weight (W)

The weight $w(v_i, v_j)$ of the edge $e(v_i, v_j)$ represents the cost to the server when a user accesses the resource node v_j from the resource node v_i. This cost is obtained as follows: probing on the website is triggered to obtain the actual access time of visiting the webpage v_j from a previously visited webpage v_i by following a link from the directed graph. The access time is from the moment a client sends a resource request to the moment a client finishes

receiving the resource. We denote the access time as response time $T(v_i, v_j)$. According to $T(v_i, v_j)$, the attacker infers the requirement of the service resources and bandwidth resources consumed by the request. When a webpage is accessed, a longer response time requires more service resources and bandwidth resources, leading to a larger server cost. So, the response time is treated as the cost to the server when a user accesses v_j from v_i. Response time varies from peak traffic hours to off-peak traffic hours because of the influence of various factors, such as the number of concurrent users, bandwidth, and processing time in the server. We repeatedly request each resource at different time and then calculate the average response time $T_{ave}(v_i, v_j)$. We can assign $w(v_i, v_j) = T_{ave}(v_i, v_j)$.

3.2 Top M Longest Path Algorithm ($TMLP$)

3.2.1 Problem description

If the weight of each edge is regarded as a distance, then our goal is to find the top M longest paths on the weighted directed graph G.

The weight of an edge represents the cost to the server when a user accesses a resource node from the previous node. If an attacker targets the consumption of the computing or bandwidth resources of the server, then the attacker can choose a vulnerable path with large total weights.

Furthermore, the attacker will avoid detection with a single attacking path and prefer to use M candidate paths with random choice. Therefore, the top M longest paths should be identified.

3.2.2 Problem transformation

In graph theory, the longest path problem is NP-hard [21]. Therefore, finding M longest paths is also a NP-hard problem and reasonably sub-optimal algorithms should be used.

The weighted directed graph G models the relationship of web-resource nodes. Considering that a resource node ultimately links back to itself, the directed graph G is one with a possible circle. The longest distance between two nodes does not have an upper bound in such a graph.

However, for a website, Beitollahi et al. [22] concluded that when users browse a website, only 10% of users have a hyperlink deeper than three levels and less than 5% of users proceed in the hyperlink of the website deeper than four levels. The “hyperlink depth” is the number of sequential-interlink pages requested by a user and is an effective attribute for the attackers. Therefore, an attacker who mimics the browsing behavior of a normal user should exploit the longest path with limited hyperlink depth L.

Paths of repeated nodes appear because the resource nodes are linked back to themselves. If a path with many repeated resource nodes is used for an attack, it is easily detected. Therefore, the repeated times (h) of nodes in a path are limited in our model.

According to the above analysis, we can simplify the problem as follows: in the weighted directed graph G with circle, top M longest paths with no more than L nodes are identified. The longest paths we obtained may have duplicate nodes. However, the repeated times of a node should be less than h.

3.2.3 Top M longest path algorithm

A dynamic-programming algorithm is used to compute the top M longest paths with a polynomial-time complexity.

Suppose that the graph $G = (V, E, w)$ is represented by an adjacency matrix $W = (w_{ij})$. According to the above analysis, we limit any candidate path to be no more than L nodes. p_{ij} is denoted as a longest path from node i to node j with no more than L nodes. The path p_{ij} contains at most $L-1$ edges.

For the nodes i and j, we decompose path p_{ij} into $i \xrightarrow{p_1} q \to j$, where path p_1 now contains at most $L-2$ edges. As the proof is shown in the following, p_1 is a longest path from i to q with no more than $L-2$ edges. $d_{ij}(l)$ is the maximum distance of any path from node i to node j that contains at most l edges. We recursively define $d_{ij}(l) = Max(d_{iq}(l-1) + w_{qj})$. q is any node that leads to j.

Proof: If path p_{ij} is decomposed into $i \xrightarrow{p_1} q \to j$, where path p_1 now contains at most $L-2$ edges, we have the distance $dist(p_{ij}) = dist(p_1) + w_{qj}$. Now, we assume a path p_2 from node i to node q with no more than $L-2$ edges with distance $dist(p_2) > dist(p_1)$. Then, $i \xrightarrow{p_2} q \to j$ is a path from i to j whose distance $dist(p_2) + w_{qj}$ is more than $dist(p_{ij})$, which contradicts the assumption that p_{ij} is the longest path from i to j with no more than $L-1$ edges. Then, p_1 is the longest path from i to q with no more than $L-2$ edges.

The key idea of the algorithm is implemented in Algorithm 1, where, n is the number of the nodes and W is the adjacency matrix. To limit the occurrence times of a specific node on a path, we add the array t for checking the occurrence times of each node on a path. The repeated times of nodes are limited to h.

We use a heap to sort the longest paths. After the dynamic-programming algorithm, we scan the array d and select the value of the top M longest paths.

If s_{ij} is defined to be the predecessor of node j on a path from node i to node j, then the predecessor of node j is obtained by s_{ij}. Therefore, as shown on the Algorithm 2, we construct the longest path from i to j by recursively calling function FindPath(i, j).

Algorithm 1 Longest Path Algorithm

LongestPath(L)

```
for i ← 1 to n do
  for j ← 1 to n do
    for l ← 2 to L − 1 do
      d_ij(l) ← 0
    end for
    d_ij(1) ← T_ave(i) + w_ij
    s_ij ← −1
    t_ij(i) ← 1
    t_ij(j) ← 1
  end for
end for
for l ← 2 to L − 1 do
  for q ← 1 to n do
    for i ← 1 to n do
      for j ← 1 to n do
        if (d_iq(l − 1) + w_qj > d_ij(l))&&(t_ij(q) < h) then
          d_ij(l) ← d_iq(l − 1) + w_qj
          s_ij ← q
          t_ij(q) ← (t_ij(q) + 1)
        end if
      end for
    end for
  end for
end for
return d and s
```

Algorithm 2 Constructing Longest Path

FindPath(i, j)

```
if (s_ij! = −1) then
  FindPath(i, s_ij)
  Print j
end if
```

3.2.4 Analysis of algorithm

a) Time complexity: Algorithm 1 has four nested loops, where the three loop index (q, i and j) take on at most n values, and another loop index l takes on at most $L-2$ values. Hence, the running time of the algorithm is $O(n^3 \times L)$.

b) Space complexity: The algorithm is optimized by using a temporary array r. At the start of each round of the dynamic-programming, the value of the array r is assigned as zero. In dynamic-programming, the formula is $r_{ij} = \max(r_{ij}, d_{iq} + w_{qj})$. At the end of each round of dynamic-programming, we assign the optimal value r to d; hence, the algorithm requires $O(n^2)$ space to store the array d and s. In our illustrative study, the space required is about 50 MB.

4. Path Vulnerabilities-Based (PVB) Attack and Protection

4.1 PVB Attack

PVB attack is a new kind of DDoS attack where an attacker creates a set of vulnerable paths (D) of a targeted site and randomly chooses one of the vulnerable paths to attack.

In this attack strategy, a hacker first analyzes the contents of web pages of a website, incurs probing by accessing webpages or hyperlinks, measures the access time from one page to another page, and then uses the previous information to construct a weighted directed graph and model the relationship of web pages in the website. The hacker uses the Top M Longest Path algorithm to find the Top M longest paths. As previously discussed, a longer traversed path consumes more resources. Hence, the path is regarded as a vulnerability that can be used to trigger the asymmetric attacks of the website. Selecting the number of paths to attack affects the result of the attack and the difficulty of detection. If the number is small, then the attack effect will be more efficient but easily detected.

PVB attack is more effective and difficult to detect than a random attack. PVB attacks websites with numerous direct links to pages on the same server, such as news and education websites, especially those websites maintained by organizations themselves. PVB attacks can destroy these web servers with minimal attack traffic while simultaneously eluding common detection mechanisms.

4.2 Defense Model against PVB Attack

A PVB attack is detectable if its mechanism is well understood. A webserver can maintain a receiving window of request for each session to effectively detect a PVB attack. For a client IP, its browsing behavior is modeled as a request sequence. v_i denotes the request i in a request sequence. $time_i$ denotes the timestamp of request i. As shown on **Fig. 1**, if $time_i$-$time_{i-1} \le t$ (t is set to a default value of 30min), then v_i and v_{i-1} are treated as in the same session; otherwise, they are treated in separate sessions.

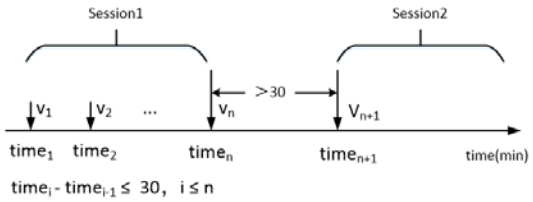

Fig. 1. Definition of session

Hence, a session is defined as Formula (1).

$$session \quad S_k = \left\langle ip, \left\{ \left\langle v_1, time_1 \right\rangle, \left\langle v_2, time_2 \right\rangle, \cdots \left\langle v_n, time_n \right\rangle \right\} \right\rangle \qquad (1)$$
$$where \quad time_i - time_{i-1} \le t, t = 30 \text{ min}, \ i \in \{1, n\}$$

Many differences can be observed between the session of normal users and attackers. In our detection strategy, we consider four factors: repeat times of a request sequence, average response time, average popularity, and average transition probability of the request resource.

These factors are explained as follows:

(1) Repeat times of the request sequence: when launching a PVB attack, an attacker exploits the security holes and vulnerabilities of the agent machines and plants the vulnerable computers with an attack code. To protect the attack code from discovery, the agent machines assign only a small subset of those attack paths. Based on the above assumption, the agent machines should repeatedly request the few assigned paths to effectively achieve the attack. Hence, this factor is useful to detect PVB attack.

(2) Average response time of the requested resources: as mentioned above, the response time $T_{ave}(v_i, v_j)$ is the weight $w(v_i, v_j)$ of the edge $e(v_i, v_j)$ on directed graph G. Based on the weighted directed graph G, an attacker chooses the vulnerable paths with large total weights to launch a PVB attack. $w(s_k)$ denotes the average response time of the resources in session s_k and is calculated by the following formula:

$$w(s_k) = \frac{T_{ave}(v_1) + w(v_1, v_2) + w(v_2, v_3) + \cdots + w(v_{n-1}, v_n)}{n} \tag{2}$$

When a PVB attack is launched, the average response time of resources in attack sessions is larger than that in normal sessions. Hence, this factor is effective for the detection of PVB attack.

(3) Average popularity of the requested resources: web resources have different access frequencies on a website. Jung et al. [23] observed that 10% of webpages account for approximately 80%-90% of requests. The popularity of resource v_i is defined as follow:

$$P_{v_i} = \frac{Access_{v_i}}{Access_{all}} \tag{3}$$

Access_{v_i} is the number of times the resource v_i is accessed, whereas $\mathrm{Access}_{\mathrm{all}}$ is the number of times all the resources are accessed in this period of time without any attack. $P(s_k)$ denotes the average popularity of resources in a session s_k and is calculated by the following formula:

$$P(s_k) = \frac{P_{v_1} + P_{v_2} + \cdots + P_{v_{n-1}} + P_{v_n}}{n} \tag{4}$$

When a PVB attack is launched, the average popularity of accessed resources in the attack sessions deviates from that in normal sessions.

(4) Average transition probability of two adjacent requests: the transition probability between every two resource nodes is different in a website. This probability is calculated by the following formula:

$$Tr(v_i, v_j) = \frac{Tr_{i \to j}}{Tr_{i \to all}} \tag{5}$$

$Tr(v_i, v_j)$ represents the transition probability from the resource node v_i to the resource node v_j. $Tr_{i \to j}$ is transition times from resource v_i to resource v_j, and $Tr_{i \to all}$ is the transition times from resource v_i to all resources. $Tr(s_k)$ denotes the average transition probability of all adjacent requests in a session s_k and is calculated by the following formula:

$$Tr(s_k) = \frac{Tr(v_1, v_2) + Tr(v_2, v_3) + \cdots + Tr(v_{n-1}, v_n)}{n-1} \tag{6}$$

When a PVB attack is launched, the average transition probability of all adjacent requests in attack sessions is escalated from that in normal sessions.

4.2.1 Finite-state Machine Model

We use the finite-state machine (FSM) [24] to model the dynamic processes for the state of the user's session and monitor the PVB attacks.

In our model, FSM is specified by (Q, q_0, V, f, O, h), where

- Q is the set of states of user's session;
- q_0 is the initial state, an element of Q;
- V is the set of input of the session;
- f is the state-transition function;
- O is the set of output of the session;
- h is the output function used to calculate the set of output of the session.

In the following, each of these components is described in detail to model the dynamical processes for the state of the user's session.

(1) States (Q): Q is the set of states of user's session: $Q = \{q_0, q_{v_1}, q_{v_1v_2} \cdots q_{v_1v_2 \cdots v_i} \cdots\}$, which represents the different states of the user's session. q_0 is the initial state of Q. Each state represents a different request sequence. For example, q_{v_1} represents the request sequence $\{v_1\}$ and $q_{v_1v_2,\cdots v_i}$ represents the request sequence $\{v_1,v_2,\ldots,v_i\}$.

(2) Input (V): V is the set of input of the system: $V = \{v_1, v_2, v_3, \cdots, v_n\}$.We maintain a receiving window of request for each session. V is the set of the resource nodes requested in a session.

(3) State-transition Function (f): the next value of the state $q[t+1]$ is obtained by the state-transition function $f(v_\theta, q[t])$, where $v_\theta(\theta \in [1, n])$ is the current input and $q[t]$ is the current state. State-transition function is implemented by a set of rules. For example, if the current state is q_{v_1}, that represents the request sequence $\{v_1\}$, the

current input is V_2, and $V_2 \neq V_1$, then it transfers to state $q_{V_1V_2}$, which represents the request sequence $\{V_1, V_2\}$. If $V_2 = V_1$, then it stays in state q_{V_1} and the occurrence times of the state may add one. The state diagram is shown in **Fig.2**, and the rules are formulated as follows:

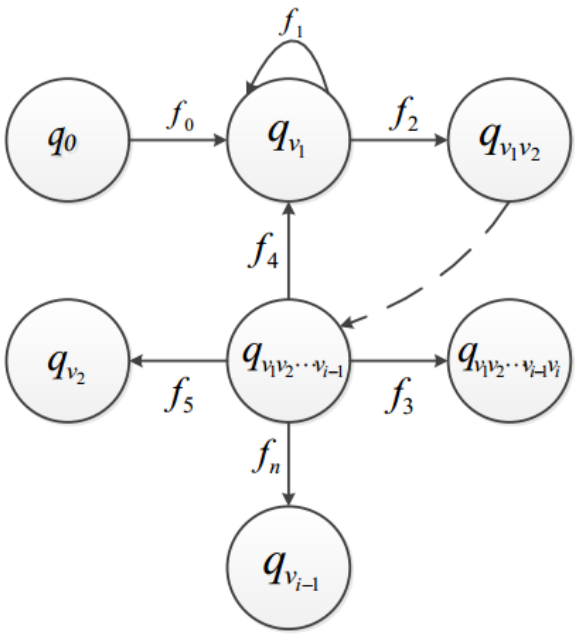

Fig. 2. State diagram of the FSM for the user's session

f_0 : **IF** $(q[t]\, is\, q_0)$ **AND** $v_1 \notin \{\varepsilon\}$
Then $(q[t+1]\, is\, q_{v_1})$ **AND** $(N[t+1](q_{v_1}) = N[t](q_{v_1}) + 1)$

f_1 : **IF** $(q[t]\, is\, q_{v_1})$ **AND** $v_2 = v_1$
Then $(q[t+1]\, is\, q_{v_1})$ **AND** $(N[t+1](q_{v_1}) = N[t](q_{v_1}) + 1)$

f_2 : **IF** $(q[t]\, is\, q_{v_1})$ **AND** $v_2 \notin \{v_1\}$
Then $(q[t+1]\, is\, q_{v_1v_2})$ **AND** $(N[t+1](q_{v_1v_2}) = N[t](q_{v_1v_2}) + 1)$

f_3 : **IF** $(q[t]\, is\, q_{v_1v_2\cdots v_{i-1}})$ **AND** $v_i \notin \{v_1, v_2, \cdots, v_{i-1}\}$
Then $(q[t+1]\, is\, q_{v_1v_2\cdots v_{i-1}v_i})$
AND$(N[t+1](q_{v_1v_2\cdots v_{i-1}v_i}) = N[t](q_{v_1v_2\cdots v_{i-1}v_i}) + 1)$

f_4 : **IF** $(q[t]\, is\, q_{v_1v_2\cdots v_{i-1}})$ **AND** $v_i = v_1$
Then $(q[t+1]\, is\, q_{v_1})$ **AND** $(N[t+1](q_{v_1}) = N[t](q_{v_1}) + 1)$

f_5 : **IF** $(q[t]\, is\, q_{v_1v_2\cdots v_{i-1}})$ **AND** $v_i = v_2$
Then $(q[t+1]\, is\, q_{v_2})$ **AND** $(N[t+1](q_{v_2}) = N[t](q_{v_2}) + 1)$

...

f_n : **IF** $(q[t]\, is\, q_{v_1v_2\cdots v_{i-1}})$ **AND** $v_i = v_{i-1}$
Then $(q[t+1]\, is\, q_{v_{i-1}})$ **AND** $(N[t+1](q_{v_{i-1}}) = N[t](q_{v_{i-1}}) + 1)$

Where N denotes the occurrence times of a state. For example, $N[t](q_{v_1})$ is the current value of the occurrence times of the state q_{v_1}. $N[t+1](q_{v_1})$ is the next value of the occurrence times of the state q_{v_1}. The initial value of N is 0.

(4) Output (O): O is the set of output of the system. A session has three outputs: $\{O_0 : Security, O_1 : Suspect, O_2 : Dangerous\}$.

(5) Output Function (h): We obtain the value of output $O[t]$ using the output function $h(N(q[t]), w(q[t]), P(q[t]), Tr(q[t]))$ where $q[t]$ is the current state.

a) $N(q[t])$ denotes the occurrence times of the current state $q[t]$.

b) $w(q[t])$ denotes the average response time of the resources in current state $q[t]$. For example, if $q[t]$ is $q_{v_1v_2\cdots v_{i-1}v_i}$, then $w(q[t])$ is calculated by the following formula:

$$w(q[t]) = \frac{T_{ave}(v_1) + w(v_1, v_2) + w(v_2, v_3) + \cdots + w(v_{i-1}, v_i)}{i} \tag{7}$$

c) $P(q[t])$ denotes the average popularity of the resources in the current state $q[t]$. For example, if $q[t]$ is $q_{v_1v_2\cdots v_{i-1}v_i}$, then $P(q[t])$ is calculated by the following formula:

$$P(q[t]) = \frac{P_{v_1} + P_{v_2} + \cdots + P_{v_{i-1}} + P_{v_i}}{i} \tag{8}$$

d) $Tr(q[t])$ denotes the average transition probability of the all adjacent requests in the current state $q[t]$. For example, if $q[t]$ is $q_{v_1v_2\cdots v_{i-1}v_i}$, then $Tr(q[t])$ is calculated by the following formula:

$$Tr(q[t]) = \frac{Tr(v_1, v_2) + Tr(v_2, v_3) + \cdots + Tr(v_{i-1}, v_i)}{i-1} \tag{9}$$

4.2.2 Detection Architecture

Our detection architecture is illustrated in **Fig. 3**. Detection is divided into two phases: training and monitoring.

(1) Steps of the training phase are as follows:

a) Analyze the contents of the website and build a weighted directed graph G.
b) Divide the normal user's requests into different sessions.
c) For a normal session s_k, compute the average response time $w(s_k)$, average popularity $P(s_k)$, and average transition probability $Tr(s_k)$ of the resources in the session s_k.
d) Use the normal user's sessions to train the FSM model and compute the

threshold of the FSM model.

(2) Steps of the monitoring phase are as follows:

a) Divide the user's requests into different sessions and maintain a receiving window of request for each session to obtain the input vectors for the FSM model.

b) For a state $q[t]$, compute the occurrence number $N(q[t])$ of the state $q[t]$.

c) For a state $q[t]$, compute the average response time $w(q[t])$, average popularity $P(q[t])$, and average transition probability $Tr(q[t])$ of the resources at the state $q[t]$.

d) Detect the PVB attack using FSM model.

e) Output the result based on the threshold obtained in the training phase.

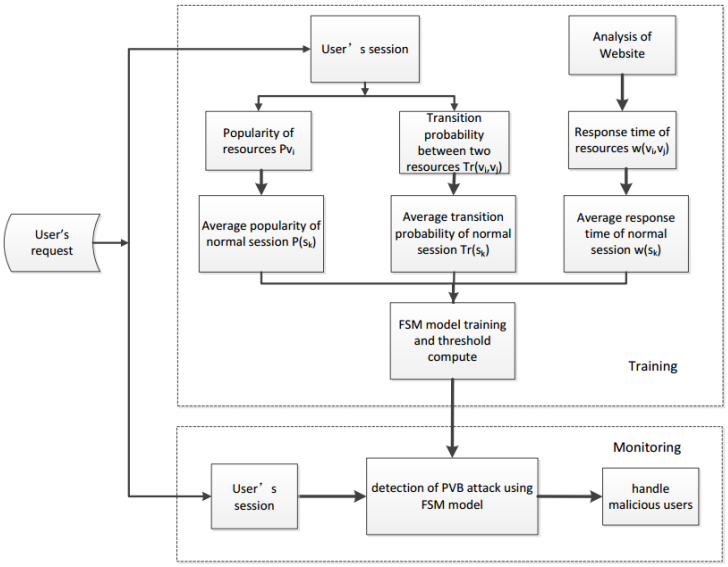

Fig. 3. Detection architecture

5. Numerical Results

The efficiency of our attack strategy was validated through real-traffic simulation experiments.

5.1 Weighted Directed Graph

Our illustrative study uses the website of Sun Yat-sen University.

The web crawler Heritrix is used to fetch the pages of the website of Sun Yat-sen University.

The Python program is used to analyze the web resources and record the properties for each resource. The program also records the link relationship between each pair of web resource nodes.

The weighted directed graph of the website of Sun Yat-sen University has 3068 resource

nodes, among which are 1835 html, 891 jpg, 101 pdf, 133 gif, 45 png, 41 bmp, and 22 other nodes.

Using several nodes as examples, we draw the directed graph of these nodes shown in **Fig. 4**. The properties of the resource nodes are shown in **Table 1**:

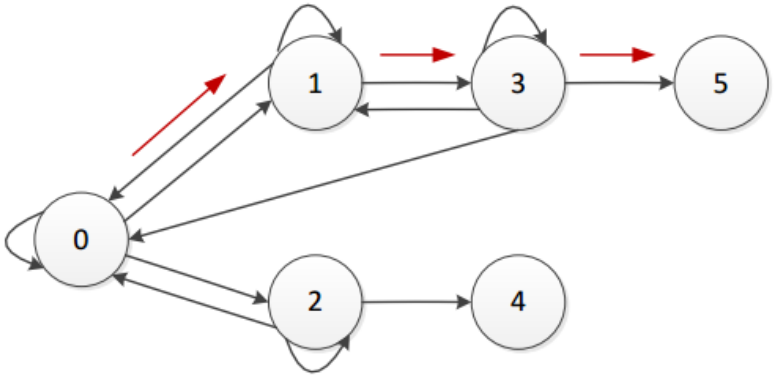

Fig. 4. A directed graph of several nodes

Table 1. Properties of the resource nodes

ID	URI	C	S(byte)
0	http://www.sysu.edu.cn/2012/cn/index.htm	html	24984
1	http://www.sysu.edu.cn/2012/en/news/news03/6082.htm	html	13391
2	http://www.sysu.edu.cn/2012/cn/zjzd/zjzd02/index.htm	html	8941
3	http://www.sysu.edu.cn/2012/en/sysu/sysu02/index.htm	html	32304
4	http://www.sysu.edu.cn/2012/docs/2012-11/20121112045538658292.jpg	jpg	4000286
5	http://www.sysu.edu.cn/2012/docs/2012-11/20121112033151248036.pdf	pdf	34429378

An experimental environment is set up to obtain the weight $w(v_i, v_j)$ of the edge $e(v_i, v_j)$. To avoid affecting the normal service of the website, the mirror http://www.sysu.edu.cn/2012/ is installed on a web server in our LAN. We access each link from the original resource to the new resource and record the response time $T(v_i, v_j)$. As previously discussed, response time varies from peak traffic hours to off-peak traffic hours

because of the influence of various factors. We repeatedly request each resource at different time and then calculate the average response time $T_{ave}(v_i, v_j)$. We assign $w(v_i, v_j) = T_{ave}(v_i, v_j)$. With **Fig. 4** as an example, the weight $w(v_i, v_j)$ is shown in an adjacency matrix:

$$\begin{pmatrix} 0.049560 & 0.008648 & 0.004038 & \times & \times & \times \\ 0.049560 & 0.008648 & \times & 0.014192 & \times & \times \\ 0.049560 & \times & 0.004038 & \times & 0.351641 & \times \\ 0.049560 & 0.008648 & \times & 0.014192 & \times & 2.975375 \\ \times & \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times & \times \end{pmatrix}$$

The Top M Longest Path algorithm is used to find its top M longest path in the above graph. For example, based on the directed graph **Fig. 4** and its adjacency matrix, if we set the parameter of the algorithm $L = 4$, $M = 1$, then the longest path can be obtained and its corresponding distance value shown in **Table 2**.

Table 2. The longest path and its distance value

Paths (L=4)	T(second)
$0 \rightarrow 1 \rightarrow 3 \rightarrow 5$	3.047775

5.2 Comparison between PVB and Random Attacks

Based on the analysis of web pages, we launch PVB and random attacks on the web server, and compare their attacking effects.

(1) PVB Attack: We set the parameter $L = 4$, $M = 6000$ to obtain the top 6000 longest paths forming the vulnerable path set D. We then randomly select 2000 vulnerable paths from the set D to launch the PVB attacks.

(2) Random Attack: We randomly request the web page of the website of Sun Yat-sen University.

5.2.1 Comparison of response traffic

The bandwidth of our web server is 1 Gbps. PVB and random attacks are launched at an identical rate of 40 requests per second. The response traffic of the web server is shown in **Fig. 5**.

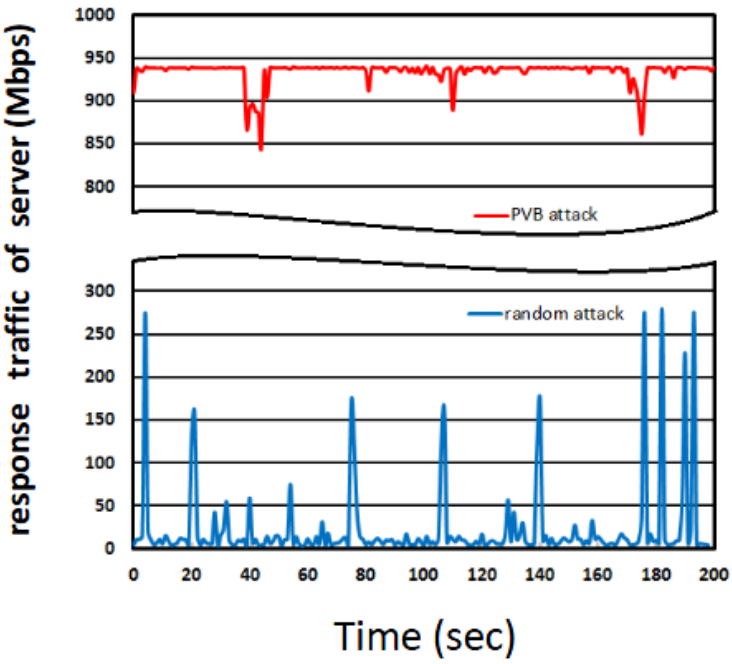

Fig. 5. The response traffic of the web server when launch PVB attack and random attack at an identical rate of 40 requests per second

From **Fig. 5**, the response traffic of the server is between 844 and 940 Mbps when we launch the PVB attack at a rate of 40 requests per second. The mean response traffic is 934 Mbps, consuming 91% of the bandwidth. The standard deviation of the response traffic is 13 Mbps. When we launch random attacks at an identical rate of 40 requests per second, the response traffic of the server is between 3 and 279 Mbps. The mean response traffic is 23 Mbps, consuming only 2% of the bandwidth. The standard deviation of the response traffic is 48 Mbps. The standard deviation is relatively large because the resources used for the attacks are randomly picked from the web site.

The response traffic for PVB and random attacks of the server are shown in **Fig. 6,** with the attack rates increasing from 1 to 40 times (1 time is 8 requests per second). The figure shows that, when the attack rate increases up to 5 times, that is, 40 requests per second, PVB attack almost filled all of the bandwidth, whereas random attack only took 2%. Given the limitations of the bandwidth, the response traffic of the server did not significantly change when the attack rate further increased with the PVB attack. When the random attack was launched, the response traffic to the attack increased along with the increase of the attack rate. However, the response traffic to random attack only took 20% of the bandwidth when the attack rate further increased to 40 times. Therefore, the PVB attack is better than the random attack. Furthermore, PVB attack can be launched at a lower request rate, thereby resulting in more difficult detection than random attacks.

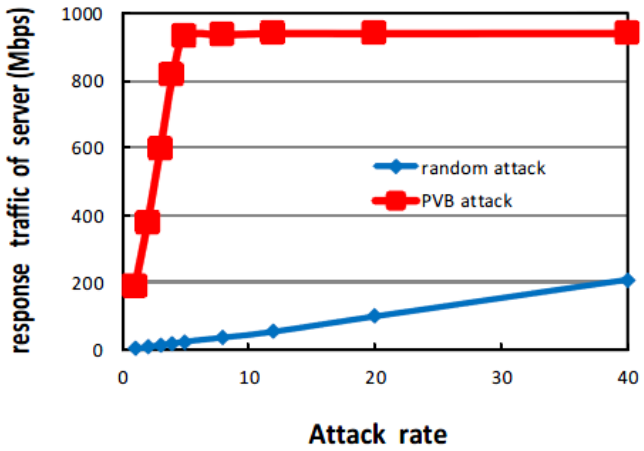

Fig. 6. The response traffic of the web server when the attack rates increased from 1 to 40 times

5.2.2 Comparison of response time

We studied and compared the average response time of 100 background users when system had additional normal sessions, random attack sessions, and PVB attack sessions respectively, increased from 0 to 300. The average response time for normal background users is shown in **Fig. 7**.

Fig. 7 shows that the average response time of the 100 background normal users slightly increased from 0.13 seconds under no additional sessions to 0.31 seconds with 300 additional normal sessions. The average increased up to 2.6 and 11 seconds with 300 attack sessions under the random and PVB attacks respectively. If the user's patience for downloading a web page is 5 seconds [10], then the PVB attack can easily drive legitimate users away from the website.

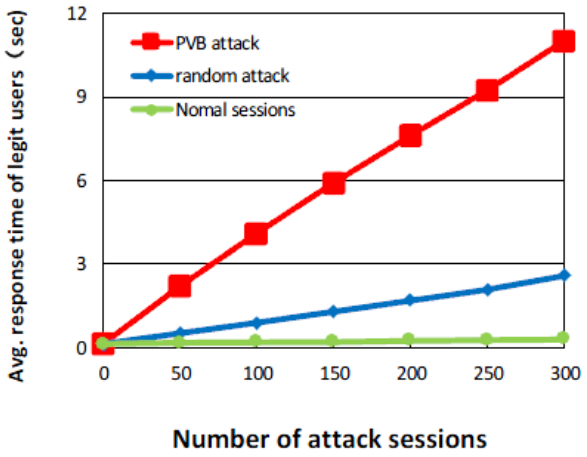

Fig. 7. The average response time for normal background users

5.2.3 Comparison of attack effect under defensive measures

A well-maintained webserver normally has some defensive measures, such as firewall and IPS. To illustrate the effect of PVB attacks under these protective mechanisms, we include the following defensive measures for a studied server:

(1) The first layer of defense is firewall. We use a Cisco adaptive security appliance (ASA) to provide access control. A Cisco ASA is a unified security device that combines firewall, network antivirus, and virtual private network capabilities, as well as provides proactive threat defense that stops attacks before spreading through the network. We limit the number of connections and download rate of each IP in access control list of Cisco ASA.

(2) The second layer of defense is intrusion prevention system (IPS). We use a Cisco intrusion prevention sensor to stop sophisticated attackers by detecting behavioral anomalies and attacks against vulnerabilities. This defense provides mechanisms to prevent attacks such as DDoS, anti-spoofing, port scanning, known vulnerabilities, pattern-based attacks and SQL injection.

(3) The third layer of defense is a software-based web-application firewall (WAF). ModSecurity installed on the web server. ModSecurity is an open source, cross platform WAF. This defense has a robust event-based programming language that provides protection from a range of attacks against web applications and allows for HTTP traffic monitoring, logging and real-time analysis [25]. In our experiment, ModSecurity is configured with the OWASP ModSecurity Core Rule Set (CRS). The OWASP ModSecurity CRS provides protections such as HTTP protocol protection, real-time blacklist lookups, HTTP denial-of-service protections, generic web-attack protections, and automation detection.

The bandwidth provided to the safety equipment is 100 Mbps, and 12 attacking computers are used to launch random attacks to the server. To fill the bandwidth of the server and make the attack effective, we increased the attack rate using the limited number of attack computers. We find that random attacks are easily detected by the system. If the attack rate of an IP is more than eight requests per second, then the server raises a warning and the IP is blocked by the server. With the increase of the attack rates from 1 to 20 times (1 time is 8 requests per second), the response traffic for random attack of the server is shown in **Fig. 8**. The figure shows that when the attack rates increased up to 14 times, all the attacking computers are found and blocked by the server.

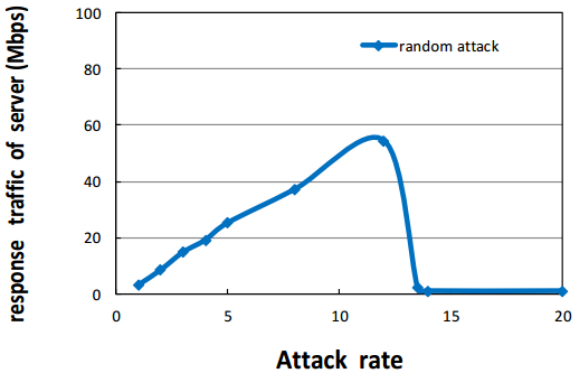

Fig. 8. The response traffic of the web server when the attack rates increased from 1 to 20 times

The same 12 attacking computers are used to launch PVB attacks to the same server. We further limit the download rate of each IP to be 10 Mbps. **Fig. 9** shows the response traffic of the server when we launch PVB attacks at a rate of 12 requests per second, that is, an attack rate of 1.5 times. The figure shows that the response traffic of the server is between 74 and 94 Mbps. The mean response traffic is 89 Mbps, consuming 89% of the bandwidth. The PVB attacks cannot be detected by these security appliances or software. PVB attacks easily take down a server with minimal attack traffic.

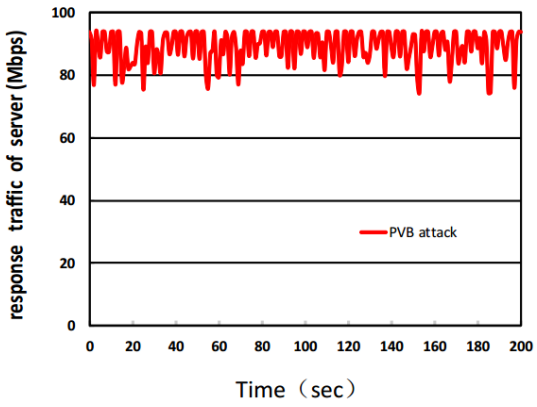

Fig. 9. The response traffic of the web server when launch PVB attack at a rate of 12 requests per second

5.3 Detection of PVB Attack

The real traffic trace log of Sun Yat-sen University was used as the original experiment data to validate our detection mechanism. The logs were collected on April 10, 2013. A total of 20978 client IP and 1,281,876 requests were included in the trace. Normal users' requests last from 0 s to 86400 s. Timestamps have one-second precision. This dataset has 2480 resources. We conducted some statistics for the dataset. In the dataset, resources have different access frequency. As mentioned above, we computed the popularity P_{V_i} of all the resources V_i. We then sorted the resources according to the descending order of the popularity and observed the cumulative popularity. As shown in **Fig. 10**, 5% of resources draw approximately 95% of web access in the website. This result is similar to most works on web mining [23]. The results follow a Zipf-like distribution.

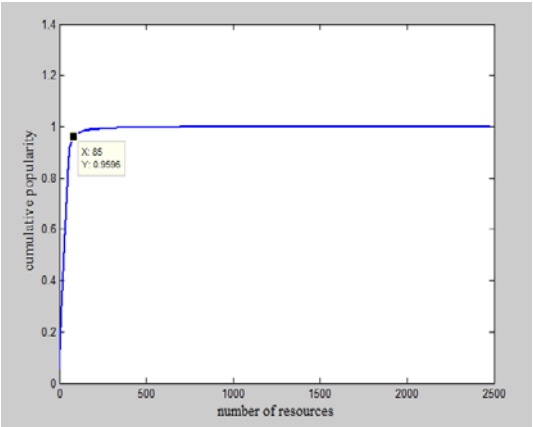

Fig. 10. Cumulative popularity of resources

To validate our detection model, we simulated the PVB attack from 40000 s to 55000 s in the trace. Our simulation used 50 attack nodes. Each node randomly selected some vulnerable paths from set D and emulated a stochastic-type pulsing attack [1]. Pulsing attacks are difficult to detect because the pulsing attackers launch attacks by periodically sending burst pulses. In our experiments, the attack period, the number of vulnerable paths chosen from the set D for attack, and the burst rate are all preset by each attack node randomly before launching the next pulsing attack. Thus, different attack nodes have different attack behaviors, whereas similar attack nodes also have different attack parameters in different attack periods. The detection mechanism based on the FSM model is used to detect the PVB attack. **Fig. 11** is the Receiver Operating Characteristics (ROC) curves showing the performance of our detection model in the PVB attack.

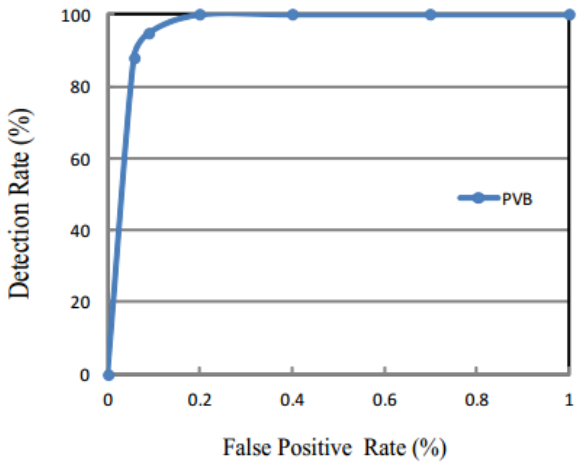

Fig. 11. ROC of PVB attack

6. Conclusion and future work

We introduced a new attack strategy named Path Vulnerabilities-Based (PVB) Attack by analyzing web pages on a website. A new algorithm called Top M Longest Path algorithm (TMLP) was proposed for automatically finding the vulnerable paths to enable the attacker to launch an effective DDoS attack. We also discussed the defense mechanism based on FSM model for the proposed PVB attacks. Numerical results showed that our attack strategy and our defense mechanism are effective. We will conduct further study on a simpler and effective detection mechanism to defend against PVB attack.

References

[1] Y. Xie and S.Z. Yu, "Monitoring the application-layer DDoS attacks for popular website," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp.15-25, February, 2009. Article(CrossRef Link)

[2] F. Simmross-Wattenberg, J.I. Asensio-Perez, P. Casaseca-de-la-Higuera et al., "Anomaly Detection in Network Traffic Based on Statistical Inference and α-Stable Modeling," *IEEE Transations on Dependable and Secure Computing*, vol. 8, no. 4, pp. 494-509, July/August, 2011. Article(CrossRef Link)

[3] N.A. Mohammed and J.R. Martin, "Uniform DoS traceback," *Computers & Security*, vol. 45, no. 3, pp. 17-26, September, 2014. Article(CrossRef Link)
[4] Y.Xiang, K. Li, and W. Zhou, "Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics," *IEEE Transactions on Information Forensics and Security*, vol. 6, no. 2, pp.426-437, June, 2011. Article(CrossRef Link)
[5] S.N. Shiaeles, V. Katos, A.S. Karakos, and B.k. Papadopoulos, "Real time DDoS detection using fuzzy estimators," *Computers & Security*, vol. 31, no. 6, pp.782-790, September, 2012. Article(CrossRef Link)
[6] H.S.Kang and S.R.Kim, "sShield: small DDoS defense system using RIP-based traffic deflection in autonomous system," *The Journal of Supercomputing*, vol. 67, no. 3, pp.820-836, March, 2014. Article(CrossRef Link)
[7] I.C.Paschalidis and G. Smaragdakis, "Spatio-Temporal Network Anomaly Detection by Assessing Deviations of Empirical Measures," *IEEE/ACM Transactions on Networking*, vol. 17, no. 3, pp. 685-697, June, 2009. Article(CrossRef Link)
[8] C.Y. Chou, B. Lin, S. Sen and O. Spatscheck, "Proactive Surge Protection: A Defense Mechanism for Bandwidth-Based Attacks," *IEEE/ACM Transactions on Networking*, vol.17, no.6, pp.1711-1723, December, 2009. Article(CrossRef Link)
[9] C. Cattani, "Harmonic Wavelet Approximation of Random, Fractal and High Frequency Signals," *Telecommunication Systems*, vol. 43, no. 3, pp.207-217, April, 2010. Article(CrossRef Link)
[10] S. Ranjan, R. Swaminathan, M. Uysal and A Nucci, "DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer Attacks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp.26-39, February, 2009. Article(CrossRef Link)
[11] V. Durcekova, L. Schwartz and N. Shahmehri, "Sophisticated Denial of Service attacks aimed at application layer," *ELEKTRO*, pp. 55-60, 2012. Article(CrossRef Link)
[12] D. Gavrilis, J. Chatzis and E. Dermatasb, "Flash Crowd Detection Using Decoy Hyperlinks," in *Proc. of IEEE Conf. on Networking, Sensing and Control*, London, pp 466-470, April 15-17, 2007. Article(CrossRef Link)
[13] J. Nazario, "DDoS attack evolution," *Network Security*, vol. 2008, no. 7, pp. 7-10, July, 2008. Article(CrossRef Link)
[14] K. Sourav and D.P. Mishra, "DDoS detection and defense: client termination approach," in *Proc. of 12th CUBE International Information Technology Conference (CUBE '12)*, pp.749-752, September 3-5, 2012. Article(CrossRef Link)
[15] D. Hayes, M. Welzl, G. Armitage and M. Rossi, "Improving HTTP performance using `stateless' TCP," in *Proc. of the 21st international workshop on Network and operating systems support for digital audio and video (NOSSDAV '11)*, pp. 57-62, June 1-3, 2011. Article(CrossRef Link)
[16] A. Raghunath, S. Ramachandran, S. Subramanian and S. Vaidyanathan, "Data rate based adaptive thread assignment solution for combating the SlowPOST denial of service attack," *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 5, pp. 1-5, September, 2013. Article(CrossRef Link)
[17] S. McGregory, "Preparing for the next DDoS attack," *Network Security*, vol. 2013, no. 5, pp.5-6, May, 2013. Article(CrossRef Link)
[18] S. Y. Nam and S. Djuraev, "Defending HTTP Web Servers against DDoS Attacks through Busy Period-based Attack Flow Detection," *KSII Transactions on Internet and Information Systems*, vol. 8, no. 7, pp. 2512-2531, 2014. Article(CrossRef Link)
[19] M.K Yoon, "Using Whitelisting to Mitigate DDoS Attacks on Critical Internet Sites," *IEEE Communications Magazine*, vol. 48, no. 7, pp. 110-115, July, 2010. Article(CrossRef Link)
[20] A. Ramamoorthi, T. Subbulakshmi, and S.M. Shalinie, "Real Time Detection and Classification of DDoS Attacks using Enhanced SVM with String Kernels," in *Proc. of IEEE Conf. on Recent Trends in Information Technology, ICRTIT*, pp. 91-96, June 3-5, 2011. Article(CrossRef Link)
[21] K. Ioannidou, G. Mertzios and S. Nikolopoulos, "The longest path problem has a polynomial solution on interval graphs," *Algorithmica*, vol. 61, no. 2, pp.320-341, October, 2011. Article(CrossRef Link)
[22] H. Beitollahi and G. Deconinck, "Tackling Application-layer DDoS Attacks," *Procedia Computer Science*, vol. 10, no. 1, pp. 432-441, 2012. Article(CrossRef Link)

[23] J. Jung, B. Krishnamurthy and M. Rabinovich, "Flash crowds and denial of service attacks: Characterization and implications for CDNs and websites," in *Proc. of 11th IEEE Conf. on World Wide Web, Honolulu, Ha-waii, USA, ACM*, pp.293-304, May 7-11, 2002. Article(CrossRef Link)
[24] A. Alvarez-Alvarez, G. Trivino and O. Cordon, "Human Gait Modeling Using a Genetic Fuzzy Finite State Machine," *IEEE Transactions on Fuzzy Systems*, vol. 20, no. 2, pp. 205-223, April, 2012. Article(CrossRef Link)
[25] SpiderLabs, "ModSecutity," https://github.com/fengxuangit/ModSecurity. Article(CrossRef Link)

Chuyu She is a Ph.D candidate in the School of Data and Computer Science at Sun Yat-sen University. She is also working as a lecturer in the School of Mathematics and Statistics at Guangdong University of Finance & Economics. Her research interests include network security and software engineering.

Wushao Wen is a professor in the School of Data and Computer Science at Sun Yat-sen University. He is co-appointed as an adjunct professor at SYSU-CMU Shunde International Joint Research Institute. His research interests include computer and network security, universal threat management, telecom networks and cloud computing security.

Quanqi Ye is an undergraduate student in School of Data and Computer Science at Sun Yat-sen University. His research interests includes: software and web security, software engineering and formal method.

Kesong Zheng is a graduate student in the School of Data and Computer Science at Sun Yat-sen University. His research interests include computer and network security, software engineering, big data, spark and data mining.