Research on Timestamp-based Port Hopping in Software Defined Network

Ziyu Zhao, Wei Huang, Yuanbo Guo & Wei Liu*
State Key Laboratory of Mathematical Engineering and Advanced Computing, Institute of Cryptography Engineering, Information Engineering University, Zhengzhou, Henan, China

ABSTRACT: Port hopping is a typical technology in moving target defense which constantly changes communication port number to confuse potential attackers and protect target hosts. In port hopping technology, time synchronization is the key problem to be solved since time synchronization can ensure communication port matching and avoid data loss in port hopping process. However, in software defined network (SDN), since its architecture differs from traditional network, port hopping technology hasn’t been fully applied in it. Based on timestamp synchronization, this paper creates a port hopping model in software defined network according to its centralized control, data plane and control plane separation as well as programmable features. In timestamp-based port hopping model, we design a three-party port synchronization mode by utilizing SDN controller and solve the problems in time synchronization such as clock drift and handoff. Our theoretical analysis and experimental results show that this proposed timestamp-based port hopping method can effectively resist DoS attack without adding load on SDN controller in software defined network.

Keywords: software defined network; port hopping; timestamp-based synchronization; handoff; Clock drift; DoS attack

1 INTRODUCTION

Since normal attacks always launch through scanning communication ports, and end ports usually expose themselves in communication links, it is convenient for attackers to obtain port information; Therefore, protecting port information against attacking is the basic principle in network defense field. In recent years, moving target defense is proposed as a proactive defense method and study in it like port hopping technology gradually appears. The main idea of this technology comes from frequency hopping. It hides communication frequency band in all available bands to reach the aim of decreasing the attack ability for interception and capture. The port hopping technology in traditional network can effectively defend DoS\textsuperscript{[1,2]} attack through changing end ports in short time slot which greatly increases the system randomness.

Compare with the adoption in traditional network, we introduce timestamp-based port hopping technology in software defined network (SDN)\textsuperscript{[3-5]} due to its centralized control feature, the data plane and control plane separation and programmability. It is easy to achieve port hopping synchronization by taking advantage of SDN controller which gets the big picture of whole network. Besides, it is convenient to adjust the timestamp-based port hopping scheme dynamically though its north API.

Through analyzing general model in traditional network, this paper proposes a new timestamp-based port hopping scheme in SDN, solves the time synchronization, clock drift and data handoff problems in the port hopping process in SDN, and greatly expands the application scope for port hopping technology.

2 RELATED RESEARCH

The key issues in port hopping technology include port pseudo-random generation and time synchronization and time synchronization is the key technology. Lee et al.\textsuperscript{[6]} propose a port hopping scheme which divide time period into several time slot and assign

*Corresponding author: 1323308566@qq.com
different port numbers for every time slot. This scheme won’t change original TCP/UDP protocol and open two ports in the port change period. G. Badishi et al.\[7\] propose an Ack-based port hopping scheme, calculate the shortest time period $\varepsilon$ from blind attack to direct attack, and propose RPH port generation algorithm. Kousaburou Hari et al.\[8\] through analyzing Ack interception situations which cause port open time prolong to $\varepsilon$ and attack situation from blind attack to direct attack propose modified RPH algorithm. Shi Leyi et al.\[9\] propose time synchronization scheme using UDP public agents, build several UDP agents to send timestamp to reach the aim of time synchronization. Zhang Fu et al.\[10\] propose HOPERRA algorithm to eliminate the negative influence of clock drift in communication synchronization and achieve time synchronization without using third party agents. Fan Xiaoshi et al.\[11\] propose dynamic clock drift algorithm which shows better performance than linear clock drift algorithm. Lin Kai et al.\[12\] propose distributed timestamp synchronization which can be adopted in large network.

To sum up, timestamp-based port hopping technology can effectively defend DoS attack. This paper combines the features of OpenFlow network which is convenient for sending and updating timestamp with port hopping technology, and proposes a general timestamp-based port hopping model. It solves the time synchronization problem in software defined network and shows efficiency in defending DoS attack.

3 TIMESTAMP-BASED PORT HOPPING MODEL IN TRADITIONAL NETWORK

In traditional network, port hopping happens in both communication parties, namely, client side and server side. Both parties negotiate with each other to generate port number and synchronize time slot. The negotiation process is as follows:

1. In the beginning, both parties authorize each other’s identity. Client side sends request message, and server side authorizes its identity. Then the server changes itself to port hopping mode and they begin to communicate after client side authorizes the server side’s identity. Port hopping begins (see Figure 1):
   1. Server side gets local timestamp and start serving;
   2. Trusted client sends timestamp request message to the server side for service timestamp;
   3. Server side sends timestamp back to the client side;
   4. Client side sends service request message to the server after generating communication port number through port pseudo-random generation algorithm;
   5. Server side sends response message after generating communication port number through port pseudo-random generation algorithm, and both parties begin communication according to negotiated port number;
   6. Server side updates timestamp and sends to client side. Both parties begin communication according to new port number;
   7. Server side completes service switch and close original port.

In traditional network, timestamp-based port hopping model involves negotiation between both parties. Server side gets local timestamp and sends it back to client side and packet sending starts after both parties negotiate port hopping number. When receiving packets, server side judges whether hopping time slot finishes. When time slots ends, the server updates timestamp and completes service switch.

Based on timestamp-based port hopping model adopted in traditional network, this paper proposes a timestamp time synchronization method in SDN, designs data handoff method and solves clock drift problem. The experiments show that the timestamp-based synchronization has good performance for defending DoS attack.

4 TIMESTAMP-BASED PORT HOPPING MODEL IN SOFTWARE DEFINED NETWORK

4.1 System overall structure

In traditional network, timestamp-based port hopping model involves negotiation between both parties.

Figure 1. Port hopping synchronization process in traditional network.

Figure 2. Port hopping system architecture in SDN.
The network topology we adopt is as follows (see Figure 2). We design two modules, namely, hopping module and time synchronization module, among controller and both communication parities. Hopping module is responsible for generating hopping number according to port pseudo-random generation algorithm. Time synchronization module aims to ensure port matching between two parties in case of packet loss and communication interruption. This paper mainly concentrates on time synchronization problem.

4.2 Timestamp-based port hopping synchronization technology in SDN

The research significance: the time synchronization method can be combined with traditional port hopping algorithm which turns out to be a proactive defense model in SDN. The programmability of north API makes the scheme easy to deploy and expand which helps to improve system performance.

The synchronization steps among controller, and both communication parties are as follows (see Figure 3):

1) Unknown host sends connecting request to the controller;
2) Controller authorizes the identity of the unknown host;
3) Controller gets its own timestamp and obtains the server’s information, sends service start instruction to the server;
4) After the server receives the instruction and timestamp, it generates new port according to port pseudo-random generation algorithm and start serving, then sends feedback message to controller;
5) After the controller receives the service-start feedback message, it sends timestamp update message to the server;
6) Server closes the original service according to the update message and completes the service switch;
7) For client, the trusted client sends service timestamp request to the controller;
8) Controller sends timestamp back to the client;
9) Trusted client calculates the service port number according to the port pseudo-random generation algorithm and sends request service message;
10) Server responds to the client.

The port hopping scheme in both parties is as follows (see Figure 4).

For client, when client needs to connect to the server, it should be authorized by controller. Trusted host receives timestamp from controller and gets port number according to the port pseudo-random generation algorithm. Then send service request and complete packet sending process in one time unit. The critical code is as follows:

```c
while(dataIsReady()){  
    unsigned int timestamp=recvCurrentTimeStamp();  
    int port;  
    getServerPort(timestamp,port);  
    setServerPort(port);  
    if(connectServer()){  
        sendData();  
    }  
}
```

For the server, when starting service, the server needs to gets timestamp from the controller, and then calculates the service port according to the port pseudo-random generation algorithm. Start network monitoring and judge whether client has packets to send. If client needs to send packet, build connection and send packets in one time slot. When sending, judge whether a time slot ends. If one time slot ends, reinitialize the process. If not, still send/receive packets. The key code in server side is as follows:

```c
unsigned int timestamp=recvCurrentTimeStamp();  
int port[3];  
getPortWindow(timestamp,port);  
setLocalPort(port[0]);  
startListening();  
while(1){  
    if(getDataRequest()){  
        recvData();  
    }  
    if(!timeUnitIsOver()){  
        timestamp=recvCurrentTimeStamp();  
        getPortWindow(timestamp,port);  
        setLocalPort(port[0]);  
        startListening();  
    }  
}
```

4.3 Clock drift synchronization algorithm in SDN

In order to eliminate the error in timestamp-based time synchronization method in one time slot, this paper adopts clock drift synchronization algorithm. Divide the nonlinear clock drift into T time slot and proceed each time slot linearly. Set the original linear time synchronization interval \(T_c = T_s\), \(T_c\) and \(T_s\) represents the time synchronization slot in the client side and the controller separately.
The client sends initialization connection request and calculates $u_1$, $u_2$ (see Figure 5). The client sends synchronization test message in $t_2$, controller receives packets and then sends response message and records time $t_3$, the client receives response message and records receiving time $t_3$. Set the clock drift rate $\Theta = \frac{T_3}{T_3}$, Get clock drift rate in $T_1$: $\Theta_1 = \frac{(t_3-t_0)}{(t_3-t_0+u_1+u_2)}$. And then set the expected clock drift rate $\Theta_2 = \Theta_1$ in the next time, and compensate the expected synchronization error in the next time slot $T_2 = T_2/\Theta_2$.

In $T_2$, still send time synchronization request and response message. Calculate the sending and receiving latency $u'_1$, $u'_2$ and calculate the real clock drift rate $\Theta_2 = \frac{(t_5-t_1)}{(t_5-t_1+u'_1+u'_2)}$. According to $\Theta_1$, $\Theta_2$, calculate the expected clock drift rate in the next time slot: $\Theta_3 = k_1\Theta_1 + k_2\Theta_2$. Here, we set $k_1 = k_2 = 1/2$.

Compensate the synchronization error in $T_3 = T_3/\Theta_3$.

Continue to calculate $\Theta_3$, and according to the formula $\Theta'_3 = k_1\Theta_{3-1} + k_2\Theta_1$ estimate the expected clock drift rate iteratively and make time compensation. After the controller make time compensation for synchronization error, controller sends update timestamp instruction to the server.

### 4.4 Communication consistency and data handoff

Data handoff can affect the system operation efficiency and the stability of the communication process. It is the key issue in the system design. Controller selects
the server from server cluster, decides the time slot period and starts the server to complete the hopping service. Data handoff happens in server change period.

For the server side: when the hopping time slot ends, controller gets its timestamp, then obtains the newly-selected server’s information, and sends its timestamp and start-service instruction to the server. The server starts hopping service according to the controller instruction and sends start-service acknowledgement back to controller. The controller sends timestamp and close-service instruction to the original server. When receiving close-service instruction, the original server obtains the newly-selected server’s port information according to the timestamp and sends socket information to the newly-selected server.

For the client side: The original server sends port information of the newly-selected server to the client and makes data migration.

After the client receives the data migration message from the original server, reinitialize the communication link to the newly-selected server and continues to send data from the data migration breakpoint, close the old link in the same time.

5 SECURITY ANALYSIS OF PORT HOPPING TECHNOLOGY

Security analysis is a key indicator to evaluate a defending method. This section mainly analyzes its defending DoS attack efficiency.

In DoS attack period, the port number won’t change in static network. In this situation, the easiest way the attacker can adopt is to scan the whole port space in sequence. Namely, when scanning number is greater than the whole port space \( n \), the attacker can discover all vulnerable ports. In static attack scenario, attacker doesn’t need to scan the port which has been scanned before, because the port won’t change. In this situation, the probability that attacker has scanned the vulnerable port number \( X \) (the whole vulnerable port number is \( v \)) in \( k \) probes depends on its hypergeometric distribution:

\[
P(X = x) = \binom{v}{x} \binom{n-v}{k-x} / \binom{n}{k}
\]  

(1)

Because the static port won’t change, it is convenient for the next probes. The successfully scanning rate for attacker scanning at least one vulnerable port is:

\[
P(X \geq 1) = 1 - P(X = 0) = 1 - \binom{n-v}{k} / \binom{n}{k}
\]  

(2)

The random change of port changes connection service port. If we select a new port to change original port from the unused port arranges, once port hopping happens, the attacker will lose all the port number he has probed. Therefore, even if the attacker can select the whole port space \( n \), he won’t find all vulnerable ports. In this situation, the probability that attacker has scanned the vulnerable port number \( X \) (the whole vulnerable port number is \( v \)) in \( k \) probes depends on its binomial distribution:

\[
P(X = x) = \binom{v}{x} p^x (1-p)^{k-x}
\]  

(3)

The probability for scanning vulnerable port number \( v \) in available port \( n \) is \( P=v/n \). The successfully scanning rate for attacker scanning at least one vulnerable port is:

\[
P(X \geq 1) = 1 - P(X = 0) = 1 - (1-p)^k
\]  

(4)

If there are 5 vulnerable ports (\( v=5 \)), the attacker can probe all the port space (\( k=n \)). In this situation, the static port won’t protect the target hosts effectively and all the vulnerable ports will be discovered.

Considering the system security level, port hopping is based on the time-stamp synchronization. Pretend the attacker finishes the blind attack period and comes to the direct attack period, the security level of port hopping can be evaluated from the below formula:

\[
R(x) = \frac{m s(l+1)t}{n}
\]  

(5)

In formula (5), \( R(x) \) represents the being-attacked port, and \( x \) represents the packets containing correct port number. \( m \) is the zombie hosts or attacker number; \( s \) is the generating packets rate; \( t \) is port hopping time slot; \( n \) is the number of all available ports; \( l \) is the time synchronization error or transmission latency. Obviously, the probability increases when zombie hosts or attacker number increase and speed up. Moreover, port hopping time slot increases will result in probability increasing. On the other hand, the probability decreases when port number increases, thus security level increasing. For example, set \( ms=20000 \) (attack rate is 19.2Mbit/s, packet length is 60 Byte), port hopping time slot \( t=0.5s \), \( l=0.2 \), \( n=65536 \), and calculate \( R(x) = 0.21 \). Therefore, the probability decreases when port number increases and port hopping time slot shortens.

In SDN, controller owns centralized control feature and connect to the source/destination switches. Therefore, all the packets go through the switches will go through controller. Compare to traditional network, the SDN architecture can achieve port hopping effectively to defend DoS attack. We can mount simple applications to authorize each equipment in SDN and judge whether the service port which the client send request message is the correct port number, thus filtering out suspicious flow and protecting the target hosts.
6 EXPERIMENT RESULTS

In order to test the availability and efficiency of the timestamp-based port hopping scheme in SDN, we use simulation software Mininet to simulate. The network topology we use is shown in Figure 2. We use NOX controller\(^{[13]}\) to achieve port generation and time synchronization.

In order to test and verify the effectiveness of the system, we use IXIA\(^{[14]}\) to inject attack flow in experiment 1 in SDN. We can find that (see Figure 6) in SDN, with the attack time increases, the data transmission rate dumps greatly. However, if we adopt the timestamp-based port hopping method, the data transmission rate enhances gradually. Timestamp based port hopping method can defend attack in large extent. From the experiments, we verify the effectiveness of the algorithm.

In order to test the defending ability further, in experiment 2 we send TCP messages to 5 hosts to test the communication status (see Figure 6). We test the data transmission successful rate in 4 blind attack tests when port number \(k=1000, 2000, 5000, 10000\) distinctively in blind attack.

![Figure 6. Anti-DoS attack data transmission results.](image)

From the above figure we can find when port number is above 1000, the port hopping model ensures above 90% data transmission and the timestamp-based port hopping method ensures the high communication quality.

Pretend that the DoS attacker has scanned vulnerable ports and starts direct attack. In experiment 3, we use SYN flooding attack way to test our model. Table 1 shows the apparent difference between the port hopping communication and the general communication.

<table>
<thead>
<tr>
<th>Attack time (Mbps)</th>
<th>Without this model</th>
<th>With this model</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.562</td>
<td>1.478</td>
</tr>
<tr>
<td>10</td>
<td>2.414</td>
<td>1.766</td>
</tr>
<tr>
<td>20</td>
<td>5.799</td>
<td>2.688</td>
</tr>
<tr>
<td>30</td>
<td>20.786</td>
<td>3.769</td>
</tr>
<tr>
<td>40</td>
<td>65.621</td>
<td>4.533</td>
</tr>
<tr>
<td>50</td>
<td>140.667</td>
<td>6.715</td>
</tr>
<tr>
<td>60</td>
<td>391.324</td>
<td>9.306</td>
</tr>
<tr>
<td>70</td>
<td>866.412</td>
<td>30.152</td>
</tr>
</tbody>
</table>

Table 1 shows the apparent difference in response time with and without port hopping method. From the experiment result, we can find that with the enhancement of the attack rate, the communication response time without port hopping method slows down sharply and down to 1000ms in 70Mbps, while with our model, the response time doesn’t slow down greatly when attack rate increases. We can see the response time is still 30ms in 70Mbps which is far beyond the communication without adopting our model.

In order to test the cost after adopting our model, we test the CPU process cost in SDN. We tests 10 times for each situation and record the highest figure. The experiment result is shown in Table 2.

<table>
<thead>
<tr>
<th>Situations in SDN</th>
<th>CPU occupancy rate without model (%)</th>
<th>CPU occupancy rate with hopping model (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>launching</td>
<td>262.3</td>
<td>278.1</td>
</tr>
<tr>
<td>10 flow entries</td>
<td>15.9</td>
<td>20.6</td>
</tr>
<tr>
<td>50 flow entries</td>
<td>28.4</td>
<td>29.8</td>
</tr>
<tr>
<td>100 flow entries</td>
<td>51.4</td>
<td>52.2</td>
</tr>
<tr>
<td>500 flow entries</td>
<td>117.2</td>
<td>131.8</td>
</tr>
<tr>
<td>1000 flow entries</td>
<td>145.8</td>
<td>159.2</td>
</tr>
</tbody>
</table>

Table 2. Highest CPU occupancy rate in various situations

7 CONCLUSION

Through analyzing the port hopping technology in traditional network, this paper proposes a timestamp-based port hopping model in software defined network, solves the key problems in port hopping process like time synchronization, clock drift and data handoff. The experiments results show that the timestamp-based port hopping scheme is easy to deploy and is effective enough. It is a lightweight port hopping scheme which can be smoothly installed in SDN.

ACKNOWLEDGEMENT

This paper is sponsored by National Natural Science Foundation of China (GN: 61309016) and the National Basic Research Program of China (973 Program) (GN: 2012CB315901, 2013CB329104).
REFERENCES


