Abstract—A key component of VoIP networks is the SIP signaling infrastructure. The reliance of public SIP servers on the Internet has opened up this critical infrastructure to a range of attacks. In particular, Denial of Service (DoS) attacks pose a serious security threat to the quality, reliability and availability of VoIP operations. In this paper, we investigate the impact of DoS attacks on SIP infrastructure, using a popular open source SIP server as a test bed. We have identified four attack scenarios that can exploit vulnerabilities in existing SIP authentication protocols, and we demonstrate the practical impact of these attacks on the target server. In response to these vulnerabilities, we have proposed several countermeasures to defend against each attack scenario. Our experimental results show that the current SIP implementation is highly vulnerable to DoS attacks and countermeasures are needed to make these servers more resilient. More importantly, we prove that authentication alone is no defence against DoS attacks in this context, and can actually increase the vulnerability of target servers instead of solving the problem of DoS attacks.

I. INTRODUCTION

VoIP (Voice over Internet Protocol) technology is gradually replacing the Public Switched Telephone Network (PSTN) as a platform for private and public telephony. During this evolution, the Session Initialization Protocol (SIP) has become a widely adopted protocol for signaling in VoIP applications [1] [2]. An important challenge for VoIP service providers is how to support security management, in order to provide a level of security that is comparable to users’ expectations from PSTN services [3]. Unfortunately, the openness of the Internet has not only brought enormous benefits but also unprecedented security challenges to VoIP applications. In contrast to the closed and centrally-controlled PSTN, any system connected to the Internet is exposed to a variety of attacks, such as denial-of-service attacks [4]. In this paper, we focus on one important aspect of security management for VoIP, namely defending against Denial of Service (DoS) attacks on the SIP signaling infrastructure.

SIP is an application-layer signaling protocol for creating, modifying, and terminating multimedia sessions with one or more participants [5]. A SIP server is responsible for interpreting incoming SIP packets and setting up sessions between callers and callees. SIP is normally run over UDP and IP protocols, which gives no guarantees of the authenticity of the source. This makes it vulnerable to many types of attacks, such as call hijacking and SIP-entity impersonation attacks. Generally, authentication mechanisms are used to address these security concerns. However, authentication is a resource-intensive process, which has the potential to be exploited as the basis for launching Denial of Service attacks.

In this paper, we focus on denial of service attacks against SIP infrastructure. Our contribution is three-fold. First, we have devised and implemented four types of CPU-based DoS attacks against a popular open source SIP server. Second, we have proposed countermeasures for each type of attack. Third, we have identified several strategies for an effective DoS attack defense mechanism and pointed out some future research directions.

The structure of this paper is as follows. Section II gives a brief introduction to the SIP protocol and its potential for DoS attacks. Section III defines four types of DoS attacks against SIP servers. These four types of attacks are then implemented and evaluated in Section IV. Furthermore, we discuss the possible countermeasures that can be used to enhance the security of the SIP server in Section V. Finally, an overview of related works in given in Section VI and conclusion is drawn in Section VII.

II. BACKGROUND ON SIP PROTOCOL

A. Introduction to the Session Initiation Protocol

The Session Initiation Protocol (SIP) has been developed by the IETF as an application layer call signaling protocol for use in VoIP services. In this section, we introduce the relevant concepts in SIP for this paper. For a detailed specification of SIP, readers are referred to [5].

1) SIP Entities: The key entities involved in SIP signaling are the User Agent, the Registrar (also known as the User Location Server), and the proxies.

- User Agent: The role of the User Agent is to initiate and receive calls on behalf of the user.
- Registrar: A server that keeps track of the location of registered users [6].
- Proxy: A signaling server that receives, processes and forwards connection requests, so that a request from a user agent reaches the desired called party.

2) SIP Messages: The signaling messages in SIP support requests from clients to Registrars or proxies, and their corresponding replies. The different categories of request messages are shown in Table I. Of particular importance to this paper is the REGISTER method, which is used by a client to request authentication by a Registrar server, so that the client’s location is maintained in the server. The corresponding reply status codes are shown in Table II.
3) SIP Operation: Each user agent is associated with a specific domain, e.g., a user is identified using a URI of the form sip:user@host, where host specifies the domain of the user. The user’s current location needs to be maintained with the Registrar for the user’s domain, so that when a call is made to that user, the proxy server of the calling party can find the current location of the desired user.

4) SIP Authentication: In order to avoid address hijacking in VoIP services, it is essential that Registrar servers authenticate users before they update their current address. The digest-based authentication process is illustrated in Figure 2. This three-step process begins with a Register request to the server. In order to authenticate the user, the Registrar server issues a challenge message, indicating that the user is currently unauthorized, and providing a nonce (a unique string that can include a time stamp) and a realm that identifies the domain of authentication. These parameters are used to generate a response to the challenge, which is based on the username, nonce and realm, and encoded using the user’s secret password. A typical authorization header is shown in Figure 1. In this way, the server can authenticate the user without requiring any state to be kept at the server. If the response is valid, then the user is successfully registered.

B. Vulnerabilities on SIP

According to [5], SIP adopts the HTTP Digest Authentication Mechanism as its standard authentication mechanism. Due to the lack of a strong authentication mechanism, SIP servers, like the SIP Registrar Service (User Location) and Proxy, must trust and process all SIP message packets by default. However, in an open network environment, such as the Internet, some SIP message packets may be generated by malicious users, whose aim is to exhaust network resources supporting the SIP-based VoIP communication service. Consequently, SIP-based systems are highly vulnerable to DoS attacks, especially when the HTTP Digest Authentication Mechanism, shown in Figure 2, is applied to the SIP protocol. Our proposed attacks are based on the HTTP Digest Authentication Mechanism.

C. DoS Attacks on SIP

We can categorize the types of DoS attacks on the SIP proxy/Registrar into three main categories: SIP Malformed Attack, Basic SIP Flood and Advanced SIP Spoof Flood. SIP Malformed Attack aims to crash a SIP server by exploiting any software vulnerabilities, such as a buffer overflow [4]. A Basic SIP Flood consists of a large number of SIP REGISTER or INVITE packets sent to targeted SIP servers to consume their bandwidth and CPU resources. There are no special requirements for the SIP packet header, which can be filled with random content. In contrast, an Advanced SIP Spoof Flood not only uses the traffic volume of a Basic SIP Flood but also has specially crafted headers, for example, the authorization header. Packets with such headers consume more CPU resources than a randomly generated SIP packet because the server needs to go through expensive authentication process to validate such carefully designed packets. In this paper, we focus on the Basic SIP Flood and Advanced SIP Spoof Flood attacks.

Generally, there are two categories of SIP server implementations: stateful servers and stateless servers. Stateful servers store the state information of each transaction in their memory. In contrast, stateless servers require the state information of each transaction to be embedded in the packet headers. Generally, in order to disrupt the target’s service, either network resources or server resources should be consumed by the attack requests. Since attacks against network resources depend more on the Internet infrastructure rather than the VoIP servers, in this paper we focus on attacks that are specially designed to consume SIP server resources.
Potentially vulnerable server resources include memory and CPU resources. During operation, when one of these two resources has been exhausted, the service of the target will be denied. A SIP server needs to keep a copy of the incoming request in its internal buffer. The size of the each request can vary from a few hundred bytes to a few thousand depending on the message type. A stateless SIP server deletes the buffered data once the message is sent out. However, a stateful server keeps the buffered data throughout each transaction or session. Obviously, stateful SIP servers are more vulnerable to memory-based DoS attacks due to the amount of information that has to be kept locally. An attacker can send many bogus SIP requests to force the SIP server to keep a large number of broken sessions [6], which can occupy all the system memory and leave no space for new SIP requests.

Stateless SIP servers are robust against memory-based DoS attacks as no state information is kept at the server. For this reason, stateless SIP servers are widely used, as they are considered to be less likely to be vulnerable to memory based attacks. However, a considerable amount of CPU time needs to be spent by stateless servers in analyzing SIP packet headers to authenticate whether the incoming packet belongs to an established session.

For example, for normal stateless SIP server operation, most CPU time is used to parse SIP messages, such as INVITE and REGISTER messages, and verify users' identities by generating and validating credentials. When the number of carefully-designed attack messages reaches a certain level, the target server’s CPU will be “busy” all the time and no new SIP requests can be served. In this paper, our main focus is to identify and analyze CPU-based DoS attacks on stateless SIP servers.

III. ATTACK METHODOLOGY

A successful DoS attack has to consume as many resources of the target server as possible. Let $R_{total}$ represent the total host resources consumed by a DoS attack, $r_{packet}$ represent the average resources consumed by each attack packet and $N$ represent the total number of attack packets. Hence, we have

$$R_{total} = r_{packet} \times N \quad (1)$$

In order to amplify the attack’s power, we can increase either the number of attacker packets or the resources consumed by each attack packet.

Based on this principle, we have designed four types of DoS attacks against SIP Registrar servers. The first type of attack is called a Basic Flood, which is purely based on traffic volume. The second type of attack is called a Static-Nonce-Based Flood, which aims to increase the resources consumed by each attack packet. The third type of attack is called an Adaptive-Nonce-Based Flood, which is designed to bypass the restriction on the validity time of a nonce. The fourth type of attack is called an Adaptive-Nonce-Based Flood with IP Spoofing, which is designed to bypass the restriction on the number of requests that can be sent from each IP address. All of these attacks exploit the authentication process on the SIP registration server. However, each differs in the level of sophistication with which it exploits the authentication process. We now describe each of these attack types in more detail.

A. Basic Flood

In this scenario, attackers continuously send a vast number of SIP requests to the target SIP server via brute force. Each attack packet is not specially designed, other than to contain repetitive or random data. As a consequence, the attack packets can be filtered by the first stage of the SIP server authentication process (shown in Figure 4a). The essence of this attack is quantity instead of quality.

B. Static-Nonce-Based Flood

In this attack scenario, each SIP request contains an Authorization Header, which includes a spoofed nonce. As shown in Figure 2, a nonce is a credential that can be used to validate a incoming request.

This attack is done in three steps. First, an attacker intercepts a nonce from the target SIP server, which can then be used to generate an Authentication Header. This interception can be done via shared media, such as ethernet or wireless. Alternatively, the nonce can also be obtained by probing the server actively. For example, an attacker sends a request containing a legitimate user ID to the target server and receives a packet with a valid nonce in return. Second, the generated Authentication Header can be used to form a SIP packet header. Finally, multiple SIP requests with such headers are sent to the target. The attack packets can only be filtered at the fourth stage of the SIP server authentication process (shown in Figure 4a). The crux of this attack is that it focuses on quality as well as quantity.

C. Adaptive-Nonce-Based Flood

Generally, each nonce has its own lifetime [7]. If the same nonce is used repetitively, it will eventually expire, which will prevent the target SIP server from processing any further requests. This limits the CPU resources that can be consumed by each such packet. Hence, the key feature in the Adaptive Nonce-based attack scenario is that the nonce is constantly refreshed before it expires.
As shown in Figure 3, this type of attack includes a component called a Nonce Refresher, which is used to probe the target SIP server constantly to obtain a valid nonce. This nonce is then used to make a spoofed authentication header, which can force the target SIP server to go through all the processes mentioned in Figure 4a. Consequently, each attack packet can consume more resources than in the previous type of attack.

D. Adaptive-Nonce-Based Flood with IP Spoofing

Some SIP servers keep track of the number of SIP requests from each individual IP address. If too many SIP requests are received from one IP address, packets from that IP address will be rate-limited or blocked. This is an effective anti-DoS measure if attack packets come from a small number of IP addresses. However, since SIP servers generally operate on top of UDP, which provides no mechanism to validate source IP addresses, it is trivial for attackers to bypass this protection mechanism by spoofing a large number IP addresses. In this attack scenario, we randomly spoof the source IP address in each request packet, which reduces the number of SIP requests from each IP address and hence bypasses this protection mechanism.

IV. EXPERIMENTAL EVALUATION

A. Experimental Setup

To evaluate the effectiveness of our four proposed attacks, we implemented the four types of SIP flood attacks described in Section III. The goal of our experiment is to test and validate the effectiveness of each SIP flood attack scenario, and find appropriate countermeasures to defend against such attacks.

As shown in Figure 5, two PCs are used as attacking machines, one PC is used as a SIP server, and one PC serves as a normal VoIP client. These four PCs are connected via a 100 Mbps Ethernet switch.

1) Client and Server: PJSUA [8] is a command line SIP user agent (UA) written using PJSIP open source SIP stack, which is installed to simulate a legitimate VoIP client.

The SIP server is a PC with linux distribution Ubuntu 6.06 [9] installed. It has 512 MB RAM and a 2.6 GHz Intel Pentium 3 CPU. A popular open source software SIP Express Router (SER) [10] is installed to provide SIP registration, proxy, and redirection services. MySQL [11] is used as a database backend to support the SER application.

2) VoIP Puzzler: VoIP Puzzler is a modified version of PJSUA that we have developed to generate the four types of attacks mentioned in Section III. As shown in Figure 6, the key aspect of the VoIP Puzzler is that it modifies the normal registration process and constantly sends VoIP Register requests regardless of the target server’s reply.

The main goal of the VoIP Puzzler is to be able to generate a large volume of attack packets with minimal overhead. However, the PJSIP library is a typical implementation based on the SIP protocol and specifies transaction-based process control on the client side, which only allows a SIP Register request to be retransmitted twice. This is unacceptable for an attack tool. Therefore, we removed state control at the transaction layer so that an arbitrary number of SIP requests such as Register, Invite, Cancel and Bye can be sent to the SIP server.

Moreover, in order to consume more server resources, any SIP request from the VoIP Puzzler needs to contain a response to the challenge sent by the server. Generally, the response is the MD5 [12] checksum of the username, the password, the given nonce value, the HTTP method, and the requested URI. To save CPU resources at the attack end, we avoid calculating the MD5 checksum for each SIP request and just reuse the response whenever possible.

Since the SIP Express Router (SER) uses a stateless nonce
implementation, the lifetime of a nonce is limited. As a result, the VoIP Puzzler has to send a REGISTER request without an Authorization Header to the SIP Registrar server to ask for a challenge packet, in which a valid nonce is included. This valid nonce can then be used to generate a SIP REGISTER request with a “valid” Authentication Header, which will force the target SIP server to consume CPU time on authenticating it.

B. Metrics for Evaluation

In our experiment, the following metrics are used to measure the resource consumption at the target SIP server.

- **CPU Usage** is used to measure CPU resource consumption, which is the main target of the attacks introduced in this experiment.
- **Flow Rate** is used to measure the bandwidth consumed by the VoIP Puzzler.
- **Required DoS Flow Rate** is used to indicate the minimum flow rate to cause denial of service. In our experiment, when CPU Usage is close to 100%, we consider that the service of the SIP server is denied.
- **User Agent Delay** is used to measure the level of disruption to a normal VoIP user.
- **Nonce Sampling Period** is used to measure the frequency with which a valid nonce is obtained from the target server.

Note that the default value for the nonce-expiry-time is 200 seconds for the following experiments.

C. Experimental Results

Figure 7 shows the CPU usage of the target SIP server under the Basic, Static-Nonce-Based and Adaptive-Nonce-Based Floods. In particular, when the packet transmission rate reaches approximately 5 MB/s, the Static-Nonce-Based and Adaptive-Nonce-Based attacks can consume almost 100% of the target server’s CPU resources, while the Basic Flood can only consume about 40% of the CPU resources. We can see that the Basic Flood is the least effective as its attack packets do not contain valid User IDs and hence the target SIP server will spend less time to deal with such packets.

Moreover, the Adaptive-Nonce-Based attack often consumes more CPU resources than the Static-Nonce-Based attack under the same attack rate as its attack packets always carry a valid nonce. However, the difference between these two attacks is generally small. This is due to the default authentication implemented at SER. As shown in Figure 4(a), the server checks whether the nonce has expired after the User ID has been checked. Generally, User ID checking is more expensive to do than nonce checking as it requires time-consuming database operations. Hence, an updated nonce does not help as the User ID checking will be done regardless.

In order to make the authentication process in the SER more robust, we need to reorder the authentication procedures. As shown in Figure 4(b), we check whether the nonce has expired before checking the user ID. In this way, the server will not spend any unnecessary resources to deal with a packet once its nonce is found to have expired.

We launched Static-Nonce-Based and Adaptive-Nonce-Based attacks on the target SIP server with the modified authentication process. The results are shown in Figure 8. Under the same 6 MB/s Static-Nonce-Based attack, 100% of the CPU resources of the original SER are consumed while only 40% are consumed for the SER with the modified authentication process. However, the performance of the Adaptive-Nonce-Based attack is unchanged for the original SER and the enhanced SER due to its refreshed nonce.

1) Effects of Nonce Sampling on Adaptive-Nonce-Based Flood: The modified SER can defeat Adaptive-Nonce-Based
Floods by tightening the nonce expiry time. However, attackers can defeat this protection mechanism by probing the target server more frequently to obtain a valid nonce and reuse that nonce.

Figure 9 shows the minimum flow rates required for Adaptive-Nonce-Based attacks to cause 100% CPU usage at the target SIP server under different nonce expiry settings. Generally, the smaller the nonce-expiry-time, the more resilient the server is to Adaptive-Nonce-Based attacks. For example, when the nonce sampling period equals 5 seconds and nonce expiry time equals 100 seconds, the Required DoS Flow Rate is 5.6 MB/s. However, under the same nonce sampling period, the Required DoS Flow Rate increases to 7.4 MB/s when the nonce expiry time equals 50 seconds and 11.5 MB/s when nonce expiry time equals 20 seconds.

Moreover, the Required DoS Flow Rate increases as the nonce sampling period increases. The larger the sampling period, the more likely it is that the nonce used in the attack has expired before it has been refreshed. Hence, as the expiry time decreases the attack becomes less powerful and requires larger volumes of attack traffic to deny service at the target. Consequently, in order to limit the effectiveness of Adaptive-Nonce-Based attacks, smaller nonce expiry times should be used by the server. Nevertheless, the Adaptive-Nonce-Based attack can be effective if the attack sources can generate a sufficient volume of requests.

2) IP Spoofing versus PIKE module: In order to limit the number of requests that will be accepted from a single IP address, the SER includes a defense module called PIKE [13], which tracks the number SIP requests received per source IP address. Once the number of SIP requests from a given IP address exceeds a pre-configured threshold, requests from that IP address will be blocked. In our experiments, all the attacks are generated from two computers. Without masquerading their source IP addresses, the SER with PIKE enabled can easily block traffic from these two machines once the attack traffic rate has reached a certain threshold. In order to defeat the PIKE module, source IP addresses need to be spoofed to simulate SIP requests from a large number of VoIP users.

To test the effectiveness of the PIKE module, we designed two types of attacks. The first type of attack is the Adaptive-Nonce-Based attack, which was described in the previous section. The second type of attack is the Spoofing attack, which is similar to the Adaptive-Nonce-Based attack except that the source IP addresses of the attack packets are spoofed. The attack traffic rate is set to be 5MB/s for both attack scenarios. However, the target SIP server has two sets of configurations, i.e., with or without the PIKE module enabled. When the PIKE module is enabled, the maximum number of SIP requests from each IP address is set to be 1,000.

While the target SIP server is under attack, a normal VoIP agent is instructed to send SIP requests to the server to test the responsiveness of the target. As shown in Figure 10, the x-axis represents the nonce expiry time of the SER, and the y-axis represents the User Agent (UA) delay in a log scale. “Baseline” represents the UA delay under normal operation, which is 1.2 ms. When the nonce expiry time equals 50 seconds, the UA delay with the PIKE module is 1.36 ms. However, it becomes 127 ms if the PIKE module is disabled. Hence, the SER using the PIKE module is effective in defeating Adaptive-Nonce-Based attacks as the UA delay is close to the Baseline UA delay. In contrast, the UA delay increases dramatically once the PIKE module is disabled.

However, when a Spoofing attack is used, the UA delay for the target using the PIKE module becomes the worst of all four scenarios in Figure 10, i.e., the PIKE module increases the vulnerability of the target server to spoofing DoS attacks. This is a surprising result, which can be explained as follows. First, Spoofing attacks can simulate attack traffic from a large number of source IP addresses, which can bypass the PIKE module. Even worse, the PIKE module adds its own overheads. In order to make sure that no single IP address has a disproportionally large number of SIP requests, an overhead is added to all requests to check the frequency of accesses from that source, which takes up CPU resources. As shown from Figure 10, when the nonce expiry time equals 20 seconds, the UA delay is 13 ms for the Adaptive-Nonce-Based attack without the PIKE module, but 124 ms for the Spoofing attack with the PIKE module enabled. Thus, adding an additional layer of defense can create the risk of creating a new source
of computational complexity for attacks to target.

The UA delay for the Adaptive-Nonce-Based attack with the PIKE module is insensitive to the nonce expiry time on the target server. This is due to the fact that once one IP address has been identified as sending too many requests, that IP address will be blocked. Hence, the server will not spend any resources to authenticate the packets from the blocked IP address. In contrast, for the other two attack scenarios, namely, the Adaptive-Nonce-Based attack without PIKE and the Spoofing attack with PIKE, no IP addresses are blocked. Consequently, the UA delay increases with the increase of the nonce expiry time, which is consistent with the observation in Section IV-C1.

V. DISCUSSION

In this section, we aim to analyze the some fundamental issues for designing a secure SIP server, and highlight several directions for future research.

A. Lightweight Authentication

From the experimental results in Section IV-C2, we can see that the authentication process can be a handicap for normal server operation. It can reduce the server’s performance by becoming a target for DoS attacks. DoS attacks are resource-consumption-based attacks, which means that any authentication process has to be lightweight. If the authentication process is expensive, the target server will be vulnerable to DoS attacks even if this authentication process can accurately differentiate legitimate packets from attack packets. A promising approach would be using a history-based approach [14] to separate malicious requests from legitimate requests. For example, if the target server keeps a database of regular caller’s IDs or IP addresses, it can block any request if the ID or IP address of that request is not found in the database when the server is under a DoS attack. This type of filter has been shown to be highly efficient as it is simple and does not need to go through complicated and expensive procedures shown in Figure 4.

B. Parameter Setting

In order to ensure that the SIP server is resilient to DoS attacks while still responsive to legitimate users, system parameters need to be tuned carefully. Generally, a small nonce expiry time can effectively prevent the nonce from being reused. However, it can also frustrate legitimate users. For instance, if a poor network connection renders a large packet round trip time, the nonce issued to the legitimate user can expire before the second request packet arrives at the server, as shown in Figure 2. The same problem applies to the threshold of the number of SIP requests from each IP address. A relatively low threshold can throttle a small scale DoS attack. However, it could also block the SIP requests from a proxy that aggregates the SIP requests from multiple SIP agents. Consequently, these parameters need to be set with great care to optimize the server’s performance. In the scenario of a VoIP conference call to a single destination, there will be a large number of SIP requests, which might trigger the detection threshold. To avoid punishing this type of application, we can use a whitelist to ensure that this kind of application will not trigger an alarm or be blocked.

C. A New Strategy for Nonce Generation

Based on the experimental results shown in Section IV-C2, we know that the same nonce can be reused by different IP addresses. However, if the server issues the nonce according to the client’s IP address, the nonce can only be used by the IP address that requests it. In our experimental setup, only two IP addresses can carry a valid nonce. Hence, the power of the Adaptive-Nonce-Based flood with IP Spoofing can be greatly reduced if such a scheme is in place. However, the disadvantage of this approach is that it will disrupt the service of normal VoIP users that use a proxy farm [7].

VI. RELATED WORK

In this paper, we identify several security threats to SIP signaling infrastructure in terms of their vulnerability to DoS attacks. We have implemented and tested four types of DoS attacks that can cripple a popular open-source SIP server. More importantly, we have also proposed several modifications of the SIP implementation that can greatly enhance the server’s robustness against DoS attacks. We also point out that strong authentication is not helpful in defending against DoS attacks.
On the contrary, it can make things even worse as authentication is generally an expensive process, which can deplete resources even more quickly under large-volume DoS attacks. Finally, we proposed several promising research directions towards solving DoS attack problems in this context.

ACKNOWLEDGMENT

We acknowledge the open source community for their generous contribution to make our work possible. We thank the anonymous reviewers for their insightful comments. This work was supported by the Australian Research Council.

REFERENCES


