Unregister Attacks in SIP

Anat Bremler-Barr Ronit Halachmi-Bekel
Interdisciplinary Center Herzliya
Email: {bremler,halachmi.ronit}@idc.ac.il

Jussi Kangasharju
Darmstadt University of Technology
jussi@tk.informatik.tu-darmstadt.de

Abstract

In this paper we present the unregister attack, a new kind of a denial of service attack on SIP servers. In this attack, the attacker sends a spoofed "unregister" message to a SIP server and cancels the registration of the victim at that server. This prevents the victim user from receiving any calls. We have tested common implementations of SIP servers and show that the unregister attack is easily performed on SIP servers which do not use authentication. Even on SIP servers with authentication, an attacker able to sniff the traffic between the client and server can still successfully attack common servers. We show that the root causes behind this vulnerability are either buggy implementations, or the SIP specification RFC which does not require sufficient security from the implementations.

We present a solution, the SIP One-Way Hash Function Algorithm (SOHA), motivated by the one-time password mechanism [6]. SOHA prevents the unregister attack in all situations. The algorithm is easy to deploy since it requires only a minor modification, namely adding one header field into the SIP messages. Furthermore, the algorithm is fully backwards compatible and requires no additional configuration from the user or the server.

1 Introduction

Voice over IP (VoIP) technology is changing the world of telephony. Instead of using the traditional, dedicated telephone network, VoIP services route all of their traffic over the commodity Internet. One of the main benefits of this is that since all the traffic passes through the Internet, the operators can offer highly price-competitive calls to their customers. However, VoIP services also suffer from the weaknesses of the Internet infrastructure. VoIP devices are easier to attack than traditional phones, which are connected to a dedicated network over dedicated lines. Attackers can use any of the usual attacks on Internet against VoIP devices. These attacks are easy and cheap to perform *because* VoIP devices are connected to the commodity Internet. People are used to voice calls having high reliability and high quality. A VoIP service that is open to many attacks might not be able to meet with these expectations, hence it is important to analyze the weaknesses in VoIP infrastructures and develop appropriate counter-measures.

In this paper, we present a new attack on a particular VoIP infrastructure, namely the unregister attack against SIP servers. The unregister attack is a denial of service attack, where a single packet sent by the attacker is enough. We show that in many scenarios this one packet is enough to prevent the victim from receiving calls. The victim in this case could be a single user, or a company. Moreover, the attacker can also divert the victim's calls to any third party, including the attacker itself. We also show that the victim would not receive any notification that she is under attack; from her point of view she appears able to receive calls, even though in reality this is not the case.

The unregister attack is based on spoofing an unregister message in the Session Initiation Protocol (SIP). SIP is a signaling protocol for Internet conferencing, telephony, presence, events notification, and instant messaging, and it is the de-facto signaling protocol for most VoIP implementations. The user can receive calls only if she is registered at the SIP server that responsible for her phone number. By spoofing an unregister message to this server, the attacker can effectively disconnect the user from the network.

In Table 1, we summarize the different scenarios of possible unregister attacks according to two main factors. The first factor is if the server uses authentication. The second factor is if the attacker has capabilities to sniff the traffic between the source and destination. Sniffing traffic by an attacker can be done in many of the deployed wireless networks.

We tested different server implementations (section 4), and found out that it is possible to launch unregis-

		Attack without Traffic knowledge	Attack with Traffic Knowledge
Server with no Authentication	Problem	Implementation bug	No covered by RFC
	Solution	Implement the RFC correctly (CSeq,Call-ID)	Our suggested algorithm SOHA
Server with Authentication	Problem	Not possible to attack	Unregister Implementation bug
	Solution	No problem	Do correct challenge check

Table 1. Summary of possible unregister attacks in the different environments

ter attacks due to bugs in the implementation of SIP in the server, or due to weak security recommendations in the RFC [10]. Moreover, the RFC does not provide any solution to the case where the server does not use authentication and the attacker can sniff the traffic. We suggest a solution (in section 5) , the SIP One-way Hash Function Algorithm (SOHA), which protects against spoofed unregister messages and is based on a similar technique of the one-time password mechanism [6, 7]. The algorithm is easy to deploy, fully backwards compatible, and does not require any prior configuration between the server and the client.

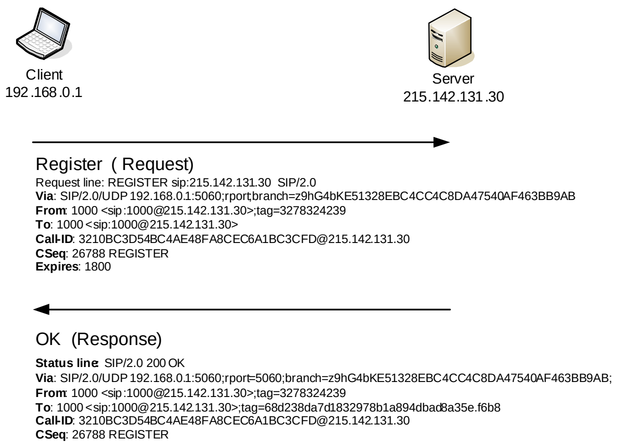
2 Background

SIP is a text-based signaling protocol used for implementing initialization, modification and termination of services that use multimedia elements like calls, instant messaging, video and presence information. SIP is a client-server protocol. Each client is registered at a server which is responsible to route the requests to the client and the requests sent from it. The client is not aware of the location of other clients it is trying to contact; the client knows only the location of the server it is registered at.

SIP can work over UDP or TCP, although it is more commonly found over UDP. Although the signaling of the call goes through the server, the media usually goes directly between the endpoints. Each endpoint publishes to the sip server the information required to send and receive media like its IP address, port and supported codecs.

There are two types of SIP servers: stateless and stateful. The stateless server does not save any information after a request or a response has been processed. This means that all the information remains in the packet. The stateful server creates a new transaction for each request it received. In both cases, the server is vulnerable to unregister attacks, since any SIP server, including stateless servers, must store the registration information.

SIP packets are composed of header fields [10], some of them mandatory and some optional, that together supply all needed information for the requested ac-



```
Request line: REGISTER sip:215.142.131.30  SIP/2.0
Via: SIP/2.0/UDP 192.168.0.1:5060;rport;branch=z9hG4bKE51328EBC4CC4C8DA47540AF463BB9AB
From: 1000 <sip:1000@215.142.131.30>;tag=3278324239
To: 1000 <sip:1000@215.142.131.30>
Call-ID: 3210BC3D54BC4AE48FA8CEC6A1BC3CFD@215.142.131.30
CSeq: 26788 REGISTER
Expires: 1800
```

```
Status line: SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.168.0.1:5060;rport=5060;branch=z9hG4bKE51328EBC4CC4C8DA47540AF463BB9AB;
From: 1000 <sip:1000@215.142.131.30>;tag=3278324239
To: 1000 <sip:1000@215.142.131.30>;tag=68d238da7d1832978b1a894dbad8a35e.f6b8
Call-ID: 3210BC3D54BC4AE48FA8CEC6A1BC3CFD@215.142.131.30
CSeq: 26788 REGISTER
```

Figure 1. Illustration of the registration flow

tion. We mention below only some of the fields that are required to understand our work: **Request-line -** holds the Request-URI (Uniform Resource Identifier), an identifier in the form of an e-mail address that indicates to which user this request stands for. **To -** specifies the recipient of a request using a display name and URI. **From -** specifies the sender of a request using a display name and a URI. **Call-ID -** is a unique identifier to all registrations of the same user for call creation and tear down. The Call-ID can be a long string (the length is not fixed), for example 32 characters, composed of numbers, letters and other characters. **CSeq -** holds a number and the method of the message. The number is incremented by one for every REGISTER message using the same Call-ID. **Proxy-Authenticate -** is an indication to the endpoint that the server requires authentication. **Expiry -** indicates how long (in seconds) the user's record will be valid for.

Our paper focuses on the registration process. The main aim of registration is to store a record in the server with the client information. When the endpoint starts, it sends a registration request. The request contains information about the IP address, phone number, name and an expiration value. If the server does not require authentication, it accepts the registration and sends a confirmation message (OK). Otherwise, the server will ask the user to identify using a challenge/response method. The client will have to answer the challenge by activating some hash function (for example, MD5) on the challenge using a password supplied to him before. The server verifies the response and if the response is as accepted the server will accept the packet (or packets).

The user's registration will be valid for the period time given in the expiry field. After the first registration, the user will send periodical registration requests according to the configured expiry value or the expiry value received from the server. Figure 1 illustrates the registration using server that does not support authentication.

The SIP server should hold up-to-date information about its clients. Therefore, when an endpoint disconnects, the server should remove its record. Endpoint removal can be done in two ways: 1. Wait for the record to expire according to its last expiry value (the common default value is 30 minutes). 2. Send an unregister message – a register message with expiry value set to zero that will cause an immediate removal.

3 Related Work

The vulnerabilities in VoIP can stem from other protocols used for VoIP servers, either VoIP-specific (e.g., SIP), or general protocol vulnerabilities (e.g., DNS or TCP). The work in [12] deals with possible DOS and DDOS attacks in SIP protocol which flood the server with specific SIP messages or misuse some of SIP features, causing the server to consume all its memory or to use all its CPU resources. In addition, the paper describes attacks that are based on other protocols used by SIP like DNS or TCP. The work in [3] covers attacks based on SIP message flows like call tear down or cancelation of a call. PROTOS [2] is a Test-Suite that tests implementations of protocols and is available also for the SIP protocol. The main tests it preforms are implementation glitches such as buffer overflow and some attacks that use the SIP INVITE message. Recently, a new set of security tools was released [5]. One of the tools is an implementation of the unregister attack in the case of a server without authentication. However, as far as we are aware, we are the first to provide a solution to the problem.

4 Unregister Attack

The RFC [10] recommends the server to support explicit unregister request (i.e., REGISTER message with expiry time zero). The motivation behind this recommendation is to allow the clients to update their location and status on the server in real time. During the update, the server is able to indicate the caller that the callee is currently unavailable and avoid sending unnecessary packets. Moreover, the server can save its own resources by releasing information about disconnected users.

However, as we show in this section, this mechanism opens new attack possibilities. The basic idea is that the attacker spoofs an unregister message and thus removes the user from the server and prevents her from receiving calls. Not only does the spoofed message disconnect the user from the server, but the user does not get any notification from the server so she will think she is still connected. In some cases an attacker can even spoof a subsequent REGISTER message, such that all the calls to the victim will be routed to the attacker or to any third party.

As we have found, the root cause of this problem is either an incorrect implementation of the RFC or that the requirements in the RFC are not secure enough concerning the unregistering of users.

4.1 Methodology

We tested the unregister attack as follows. We registered a client with different servers in different configuration (with and without authentication). We then have an attacker try to unregister the client at the server. After the attack, we try to call the victim from a third client. We repeated the test several times for each configuration to get conclusive results.

We checked three commonly used SIP servers from three different vendors. Two of them were configured in our lab without authentication (their default configuration), and one server was configured with authentication and was a server in production.

The clients were soft phones. The attacker was a small program written in C using raw sockets. The location of the attacker varied between a machine on the same LAN as the victim to locations in different networks. We did not observe any effect on the results caused by the location of the attacker.

In their default configuration, most servers are not configured to use authentication. Some possible explanations for this could be: (i) Having authentication requires account management on the server side, thus increasing the effort and complexity of managing the server. (ii) Using authentication reduces the throughput of the server. For example [11] shows that a server with authentication handled 56% - 70% less calls than the server without authentication. (iii) Authentication brings new possibilities for attacking the server. For example, [12] shows examples of DDOS attacks based on the current SIP authentication mechanism which consume all the memory and all the CPU of the server.

In many cases, authentication is used, since the server needs to know who the users are. This is required for example to do billing and accounting, since many VoIP services which use SIP are not free. To our knowledge, there are no published studies about which percentage of SIP servers use authentication and which do not. Hence, in this paper we consider both cases.

We consider two different scenarios in addition to the two kinds of servers. In one scenario, the attacker is assumed to be able to sniff the traffic between the victim and the server and in the other scenario sniffing is not possible.

The case of an attacker able to sniff packets between the client and server is motivated by the proliferation of wireless networks, where any client is able to see all the packets sent by other clients near her. If the wireless network is using encryption, the attacker must first crack the encryption before she can attack the victim. Not all wireless networks use encrypted traffic, and even when encryption is used, common protocols like WEP have been shown to be quite vulnerable to even small scale attacks. WPA-based wireless networks are more secure, but can also be broken by brute-force tools such as Cowpatty[1].

Even though many networks are encrypted, many wireless networks use no encryption at all. Examples of such networks are hot spots in public places, like coffee shops, hotels, and airports, where the configuration needed for encryption (i.e., giving out encryption keys or passwords) is not feasible. Furthermore, not all wireless cards support all encryption mechanisms, meaning that a public access point has to follow the capabilities of the "lowest common denominator". Also, consider a wired network in a company where an employee is sniffing the internal traffic in order to harm the company or attack people inside. Even though many companies take active measures against such attacks, it is still a possibility in some cases.

4.2 Unregister Problem

The ease of spoofing an unregister attack stems from being able to run SIP over UDP, which does not require a handshake before actually sending the data.[1]

The weaknesses we show below can be categorized in two groups. The first group concerns implementation errors of SIP servers or oversights in the SIP RFC [10] which allow non-secure implementations. Many of the issues identified below are stated in the RFC as "SHOULD", but we believe they should be made "MUST", in order to protect against the unregister attack. Some of the problems in this group are simply implementation bugs of the SIP servers. The second group concerns problems where the RFC proposes no solution. In this case, the problems only appear when the attacker can sniff the traffic between the client and the server (see Table 1).

Unregister Attack Without Traffic Knowledge on a Server with No Authentication: The Call-Id and CSeq headers in the unregister message should correspond to the same headers in the original register message. However, none of the servers we checked actually implement this check.

We succeeded to unregister the client easily from the server by sending a REGISTER message with zero expiry time and *random values* for the Call-ID and CSeq headers. Moreover, we succeeded in spoofing a fake registration message, posing as the legitimate client, with the name and phone number of the victim.

To perform this attack, we need only basic public information (i.e., victim's phone number and name alias) and basic information that can be retrieved by sending traffic to the server and capturing its response (i.e., client's IP address). The IP address can also be captured by establishing a call to the client, since the IP address is typically contained in the media description (done by SDP). Even if the user is behind a NAT or a VoIP-aware firewall and the attacker is in different network from the victim, there are still some fields indicating the victim's real IP address that are not changed in the packet (e.g., Via or Call-Id header). Some servers hide this information, and in this case spoofing the packet will be impossible without knowing the IP address of the user.

One possible solution to this attack would simply be to have the server check the Call-ID and CSeq headers. We believe the RFC recommendation regarding the Call-Id header should be changed from "SHOULD" to "MUST", which forces the server to check the header field. Strict checking of the CSeq-header, which is mandatory according to the RFC, could also prevent this attack. However, we believe Call-Id is the better choice, since the string in the Call-Id-header is much longer compared to the CSeq header, which is simply a sequence number. Even though it should be random, we have discovered that many implementations are relatively weak, and that the CSeq numbers are relatively easy to predict.

Unregister Attack Without Traffic Knowledge on a Server With Authentication: This attack is not possible. For every packet the server receives, it

[1]Note, that in case SIP was run solely over TCP, no spoofing is possible.

sends a "proxy authentication required" message with a nonce (a string) in it and expects a reply that contains some calculation of the nonce with a password given in advance.

Unregister Attack With Traffic Knowledge on a Server With No Authentication: In this case, the RFC does not suggest any prevention mechanism to prevent unregister attack. Implementing the RFC properly will not help because the attacker can capture the Call-Id- and CSeq-headers and hence can spoof the packet with all the required fields. In Section 5 we describe our solution to this attack.

Unregister Attack With Traffic Knowledge on a Server With Authentication: Surprisingly, even though the server uses authentication, we succeeded to unregister a client without knowing the password. This is because we were able to sniff the traffic between the client and the server. Although the authentication process itself is secure, we were able to capture both the challenge sent by the server and, more importantly, the response sent by the client. With the server using authentication, it is enough for the attacker to capture the last REGISTER-message sent by the client and take out the authorization string. If we re-used the same authorization string, we were successful in unregistering the victim from the server. In other words, we did a "replay attack" and the server which received the re-used authorization string canceled the existing registration without sending a new challenge.

This problem is not SIP-specific, as it seems to imply a bug in the server. Although the RFC does not address the re-use of the responses in the authentication, we expected that the server would not allow us to re-use a reply to a nonce and would decline our spoofed packet. Note that since the attacker is assumed to be able to capture all the traffic between the client and server, even Call-Id- and CSeq-headers would not protect against this attack. The problem can only be fixed by a correct implementation of the authentication procedure in the server.

5 SOHA algorithm

We now present our SIP One-way Hash Function Algorithm (SOHA), a solution for the situation where the attacker can capture all traffic between the client and server and the server does not use any authentication. As discussed above, the SIP RFC provides no protection against attacks in this scenario. SOHA will also protect against attacks in the other three scenarios, but in those cases, a correct implementation of the RFC, as well as a reformulation of some of the requirements, are sufficient as solutions.

The SOHA algorithm is similar to the one-time password mechanism [7, 6]. It is based on a "first come, first get"-rule, meaning that the first user to register a name and phone number is then considered to be their legitimate owner. The SOHA algorithm does not replace the authentication mechanism of SIP and therefore does not verify the identity of the user. Without authentication, we cannot protect against an attack where the attacker is able to spoof a register message before the legitimate user.[2] With our solution, however, we ensure that a correctly registered user cannot be un-registered by the attacker.

Our solution is based on a one-way hash function. The function is known to all and does not require any secret information for its operation. Hence, this solution is very light-weight and easy to deploy without any need of special configuration. The basic characteristic of a one-way hash function is that when given an input, it is easy to compute the output but when given an output it is very hard to find the original input.

In our solution, we add a header field,[3] called X-Hash-Authenticate to SIP messages. For example, when using the function SHA-1 [4] this header will be 160 bits long for every message the client sends to the server (registration, removal, call initiation and so on). Other common hash functions are MD4 [8], MD5 [9]. Figure 2 shows the details steps of our SOHA algorithms. In the first registration, the client chooses a random number x.[4] The client puts in the X-Hash-Authenticate-header a number which is equal to applying the chosen hash function h over x for n times (i.e., $h(h(...h(x))))$. The client can choose n as it wishes, but it should be large enough. Ideally, n is larger than the number of packets sent between the client and server during the time that the client is connected.

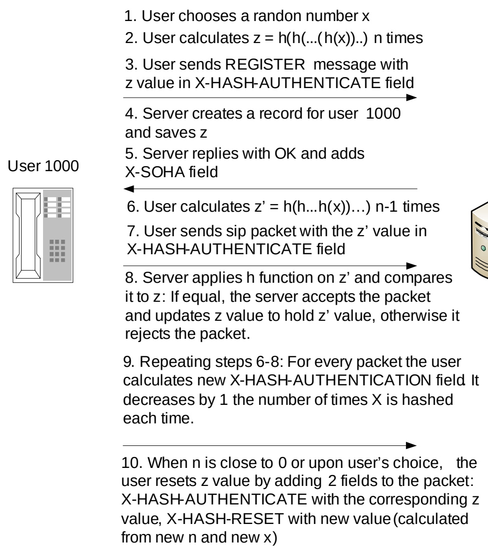
The server stores, for each client, the last value of the X-Hash-Authenticate-header it has received. In the next message, the client will send a value in this header which is equal to applying h over x for $n-1$ times. The idea is that the server can easily apply h over this value and the result should be identical to the value stored on the server from the previous header. Since the attacker cannot reverse the hash function h, she cannot find the next (i.e., previous) value, even if she manages to capture traffic between the client and server.

When n gets closer to zero, the client will send two headers in the next message. One is the X-Hash-

[2]The real owner will notice that someone has stolen his account, and can request the server owner to delete the entries manually.

[3]The SIP RFC allowed you to define your own header fields. There are actually no restrictions about the names, but the usual convention is to use a prefix X- in the new header fields.

[4]If using SHA-1, x is recommended to be a 1024 bit number



1. User chooses a randon number x
2. User calculates z = h(h(...(h(x))..) n times
3. User sends REGISTER message with z value in X-HASH-AUTHENTICATE field
4. Server creates a record for user 1000 and saves z
5. Server replies with OK and adds X-SOHA field
6. User calculates z' = h(h...h(x))...) n-1 times
7. User sends sip packet with the z' value in X-HASH-AUTHENTICATE field
8. Server applies h function on z' and compares it to z: If equal, the server accepts the packet and updates z value to hold z' value, otherwise it rejects the packet.
9. Repeating steps 6-8: For every packet the user calculates new X-HASH-AUTHENTICATION field. It decreases by 1 the number of times X is hashed each time.
10. When n is close to 0 or upon user's choice, the user resets z value by adding 2 fields to the packet: X-HASH-AUTHENTICATE with the corresponding z value, X-HASH-RESET with new value (calculated from new n and new x)

Figure 2. The steps of SOHA algorithm

Authenticate which contains the next hash value, and one is X-Hash-Reset, which contains a new hash value, obtained by picking a new random number x' and applying the hash function h for some number n' times. When the server receives the message with both of these headers, it resets its stored hash value to the one given in the X-Hash-Reset header field.

Since SOHA only adds a header field to SIP messages, it is fully backwards compatible with existing SIP implementations. In order to give the client feedback about whether the server supports SOHA, we propose that the server should add a header (X-SOHA) in its replies, so that a client is able to tell the difference between a SOHA-enabled server and a legacy server.

Alternative solution to SOHA is to solve the registration problem by using private and public keys in the client as follows. In the first packet (the registration) the client puts her public key in the message to the server. In the following messages, the client signs the data with her private key. The server can verify the signature by using the public key it has received in the first packet. The main advantage is that the signature makes the packets tamper-proof [5]. The main drawback of this solution that it is more complicated to the client and server since we need one signature operation for each packet. In the SOHA solution, the server needs to do one hash operation per packet, but hash functions are typically cheaper to compute than signatures. On the client side, we can prepare all the X-Hash-Authenticate-headers in advance. Furthermore, public keys are typically larger than hash values.

[5]In some cases a firewall or NAT may change parts of the SIP messages for security reasons . A special care should be used so the signature be only on part of the packets that are not touched by those systems.

6 Conclusion and Future Work

In the paper we have examined the unregister process in SIP protocol and found some weaknesses that can be abused by an attacker. In our future work we plan to investigate if other attacks are possible on SIP infrastructure by spoofing packets. In our preliminary tests, we found out that attacker can send spoof BYE message to an active call, and to cancel the call after call initiation. The attack is possible only when the attacker can sniff the traffic between the client and the server. We believe that our SOHA algorithm can solve also this attack scenarios.

7 Acknowledgment

The authors would like to thank deeply Zvi Gutterman. Zvi has pointed us to the private-public key solution and to the corresponding between our SOHA solution to the one time key password RFC.

References

[1] Cowpatty. http://sourceforge.net/projects/cowpatty//.
[2] Protos. http://www.ee.oulu.fi/research/ouspg/protos/-testing/c07/sip/index.html.
[3] Snocer. http://www.snocer.org.
[4] D. Eastlake and P. Jones. Sha1 rfc 3174. http://www.faqs.org/rfcs/rfc3174.html.
[5] D. Endler and M. Collier. Hacking exposed VoIP. http://www.hackingvoip.com/index.php.
[6] N. Haller, C. Metz, P. Nesser, and M. Straw. A one time password system rfc 2289. http://www.faqs.org/rfcs/rfc2289.html.
[7] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24:770–772, 1981.
[8] R. Rivest. Md4 rfc 1320. http://www.faqs.org/rfcs/rfc1320.html.
[9] R. Rivest. Md5 rfc 1321. http://www.faqs.org/rfcs/rfc1321.html.
[10] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Sip rfc 3261. http://www.faqs.org/rfcs/rfc3261.html.
[11] S. Salsano, L. Veltri, and D. Papalilo. IDMaps:the sip authentication procedure and its processing load. *IEEE Network*, 16:38–44, 2002.
[12] D. Sisalem and J. kuthan. Denial of service attacks and sip infrastructure. In *first workshop on VoIP Security in Dallas*, 2004.