IAX Voice Over-IP Security[1]

Written by Zane Lackey and Himanshu Dwivedi

Introduction

Inter-Asterisk eXchange (IAX[2]) is a protocol used for Voice-Over-IP (VoIP) communication with Asterisk servers (http://www.asterisk.org/), an open source PBX system. Along with Asterisk servers, IAX can be used between any client endpoint[3] and server system supporting the IAX protocol for voice communication.

IAX is simplistic in nature, as compared with other VoIP protocols such as H.323 and SIP. IAX uses a single UDP port between all endpoints and servers (UDP port 4569). This feature makes IAX very attractive for firewall administrators, who are often asked to open ports north of 1024 for VoIP communication. Specifically, IAX provides signaling and media transfer in-band within the protocol itself, while other VoIP implementations use multiple protocols relying on each other. For example, H.323/SIP can be used for signaling and use RTP for media transfer, which can be more confusing and complex than figuring out where the Line of Control border sits between India and Pakistan.

IAX uses a binary protocol, which claims to offers protection against buffer overrun attacks, as stated in the RFC[4]. IAX offers public key authentication with the use of RSA and call confidentiality with the use of AES. These security features are important, however, they are frequently absent in IAX deployments, leaving many IAX implementations as vulnerable as unprotected SIP or H.323 systems. Unencrypted IAX voice conversations can be sniffed, recorded, and replayed by eavesdroppers since IAX supports clear-text communication (similar to RTP media transports). The commonly used MD5 challenge response authentication mechanism specified by IAX allows passive and active adversaries to launch several attacks. These attacks include offline dictionary attacks on credentials and pre-computed dictionary attacks. Additionally, the MD5 authentication mechanism is susceptible to a middle person attack and possibly vulnerable to a downgrade attack (depending upon the implementation). The IAX draft RFC makes no recommendations for protecting against these attacks, making the authentication process a key target for malicious users. Lastly, several denial of service attacks are possible, which add to the availability concerns of IAX. These issues, along with many other concerns discussed in this whitepaper, open up the debate on whether IAX is truly a more secure alternative for VoIP communication.

For more information on the IAX architecture, see http://www.ietf.org/internet-drafts/draft-guy-iax-02.txt

---

[1] For the latest version of this paper and accompanying tools, please refer to http://www.isecpartners.com/BlackHat

[2] All references to IAX pertain to version 2 of the protocol (IAX2)

[3] Client endpoint is defined as any soft or hard phone that supports the IAX protocol

[4] See Appendix A for more information on buffer overflows and flooding attacks

The security aspects supported by IAX implementations are discussed in the whitepaper, specifically authentication, password protection, and availability. The results derived from focused research on IAX implementation, the supported authentication methods, reliability, and security features; however this paper should not be regarded as an exhaustive security analysis of the IAX protocol and its implementations. The following sections discuss each topic in detail.

1.0 Authentication Background

IAX supports three authentication methods, including plaintext authentication, MD5 authentication, and RSA authentication. In the MD5 authentication process, IAX endpoints use a challenge response system based on MD5 hashes. The MD5 authentication method protects against passive eavesdroppers who could view a clear-text password over the network as well as protection against replay attacks. The authentication scheme is vulnerable to common authentication attacks, including dictionary attacks that may allow a compromise of the endpoints credentials. The protocol also requires storage of the actual password as the password verifier[5], increasing the harm of server compromise.

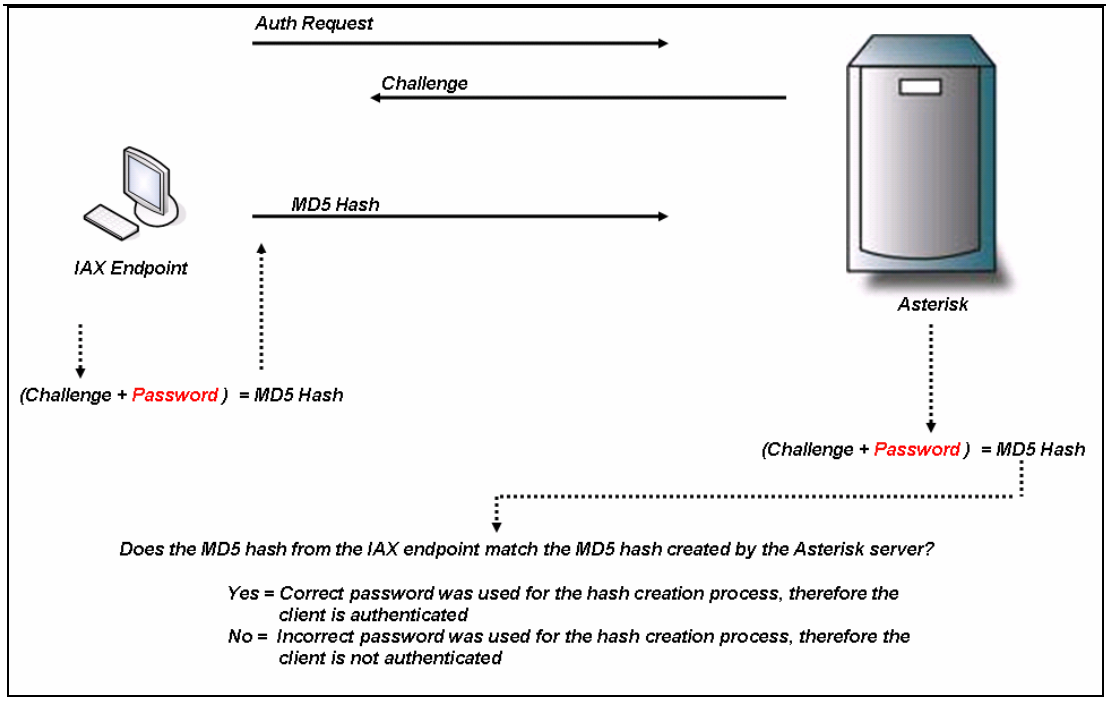
In general, MD5 allows any weak or strong password to be hashed without sending the password over the network in clear-text. For example, if an endpoint were to use the password of 'Sonia', which is a weak password since it only has 5 characters and no numbers, the MD5 hash that would be used is CCD5614CD5313D6091A96CE27C38EB22. Creating an MD5 hash of the password of Sonia ensures the password is not sent over the next in clear text; however, it exposes another problem which is the use of password-equivalent values. For example, the MD5 hash of the password of Sonia will always be the same, making it vulnerable to a replay attack. An attacker could simply sniff the MD5 hash over the network and replay it, allowing them to be authenticated instantly. The attacker does not need to know what the real password is, but simply capture the MD5 hash (the password equivalent value) and sent it over to the authenticating device. Furthermore, to speed-up the process, the attacker could simply create a MD5 hash for every word in the dictionary (pre-computed attack) and send those values to the authenticating device. While the attacker would not know the correct password, eventually they would send a MD5 hash that matches that hash for the correct password.

In order to prevent replay attacks, the challenge response method is supported by IAX. The MD5 authentication used by IAX (and other Challenge-Response solutions) does not require the use of a password or a replayable password equivalent. The challenge response method of authentication is where the authenticator, such as an Asterisk server, sends a challenge to the endpoint for each unique authentication request. For example, if an IAX endpoint tried to authenticate 5 different times, they would be given 5 different challenges, one for each authenticating attempt. Once the endpoint receives the challenge from the authenticator, the endpoint will concatenate the challenge with its password and create a MD5 hash. This MD5 hash, which is now derived from the challenge from the authenticator and the endpoint's

[5] Password verifiers are the data that must be stored in order to authenticate a peer. Ideally password verifiers are not passwords, or password equivalents, and attackers who can read the verifier store gain only the ability to launch dictionary attacks on passwords with tools like "Crack".

password, is sent over the network to the authenticating device. The authenticating device, also knowing the challenge and password, will create an MD5 hash using the challenge they produced for the endpoint and the correct password. If the MD5 hash generated by the authenticator matches the MD5 hash sent over the network by the endpoint, then the authenticator knows the correct password was used by the endpoint. If the MD5 hash sent over the network by the endpoint does not match the one created internally by the authenticating device, then the authenticator knows the correct password was not used (and the endpoint is not successfully authenticated). Figure 1.1 is an example of the IAX authentication process:

Figure 1.1



The challenge/response method prevents replay attacks, as a unique challenge is used for every authentication request. If an attacker sniffs the authentication of an endpoint, the attacker will not be able to replay the response as the challenge used to create that hash was only valid for that unique authentication request. Hence, the attacker would be trying to replay a MD5 hash that was created with a challenge tied to another session , which is therefore useless.

The following sections describe a few of the authentication weaknesses identified by iSEC, including protocol weaknesses and implementation weaknesses:

- Protocol Weaknesses: Section 1.1.1, 1.1.2, and 1.1.3
- Implementation Weaknesses: Section 1.2.1

1.1.1 Offline Dictionary Attack

The IAX MD5 authentication method protects against passwords being exposed in clear-text and against the replaying of intercepted authentication; however, it is still vulnerable to some common authentication attacks. These attacks include an offline dictionary attack that allows the credential compromise when insufficiently strong passwords are in use.

As shown in figure 1.1, a challenge is sent by the Asterisk server to the IAX endpoint over the network. This challenge is a component used to create the MD5 authentication response that is also sent over the network. Both the challenge and the response that is included in authentication process are available to a passive adversary listening on the network. While the authentication hash is not useful for direct replay, it can be used in conjunction with the challenge used to create it. Unlike an online brute-force attack, where an attacker would attempt to authenticate using a guessed passwords over and over (before being locked-out) the offline dictionary attack allows an attacker to quickly try passwords computationally on their own system, checking if they are correct without access to the targeted system. For example, if a person knew how to count, but did not know how to add, and wanted to solve the problem of 8 + x = 15, she would only need 7 attempts (1 thru 7) before brute-forcing the correct answer. The same idea applies to an offline dictionary attack. If an attacker knows the challenge sent by an attacker is 214484840 and the resulting MD5 hash is fc7131a20c49c3d96ba3e2e27d27, she can test any given password by concatenating the password with the challenge and computing the MD5. If the result is equal to the hash the attacker sniffed over the network, the attacker has guessed the correct password. See figure 1.2 and table 1.3 for more details.

Figure 1.2

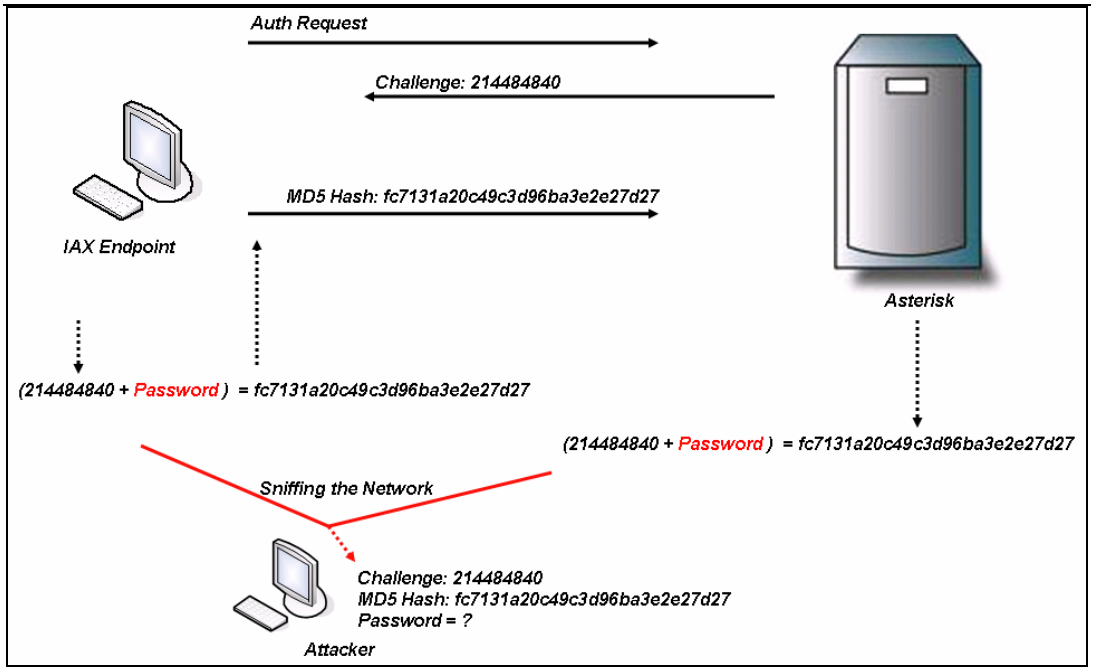


Table 1.3 Passive Dictionary Attacks

Hash = (Challenge + Password) MD5

Sniffed Information:

Challenge: 214484840

HASH: fc7131a20c49c3d96ba3e2e27d27

```
(214484840 + Hello) MD5                    = c15bc53e22ea97bc0cdfdab7754d5001
(214484840 + My)    MD5                    = b29419fd0def440dbabb364449ee067b
(214484840 + Name)  MD5                    = 7132e2f976258149318 38e63f1b05197
(214484840 + Is)    MD5                    = 6e36348e6963f75b7c18722b2ab6c0d3
(214484840 + Sonia) MD5                    = 1beb71d8a6e7a968cf6118e8b328b319
(214484840 + My)    MD5                    = b29419fd0def440dbabb364449ee067b
(214484840 + Voice) MD5                    = d63bfd9e48b2ea84c067848350f040b3
(214484840 + Is)    MD5                    = 6e36348e6963f75b7c18722b2ab6c0d3
(214484840 + My)    MD5                    = b29419fd0def440dbabb364449ee067b
(214484840 +Passport)MD5                   = 140611b567a289b074bdcd43494b92f9
(214484840 + 123voiptest) MD5              = fc7131a20c49c3d96bf69ba3e2e27d27
```

Notice the last row in table 1.3, where the generated MD5 hash matches the sniffed MD5 hash capture over the network. This information allows the attacker to verify they have identified the correct password, which was "123voiptest". Furthermore, unlike other password attacks, the attacker only needs to capture a challenge and MD5 hash once to perform this attack. The challenge will always be valid for the MD5 hash sniffed over the network, which all the information required to perform the passive attack.

iSEC has released a proof of concept tool called IAX.Brute[6], which automatically performs a passive dictionary attack with an intercepted challenge and corresponding hash from a valid authentication response. iSEC's testing shows a dictionary file of 280,000 words will take less that one minute to test on a Thinkpad T42 with an Intel PIV processor (running 1.83GHz).

1.1.2 Active Dictionary Attack

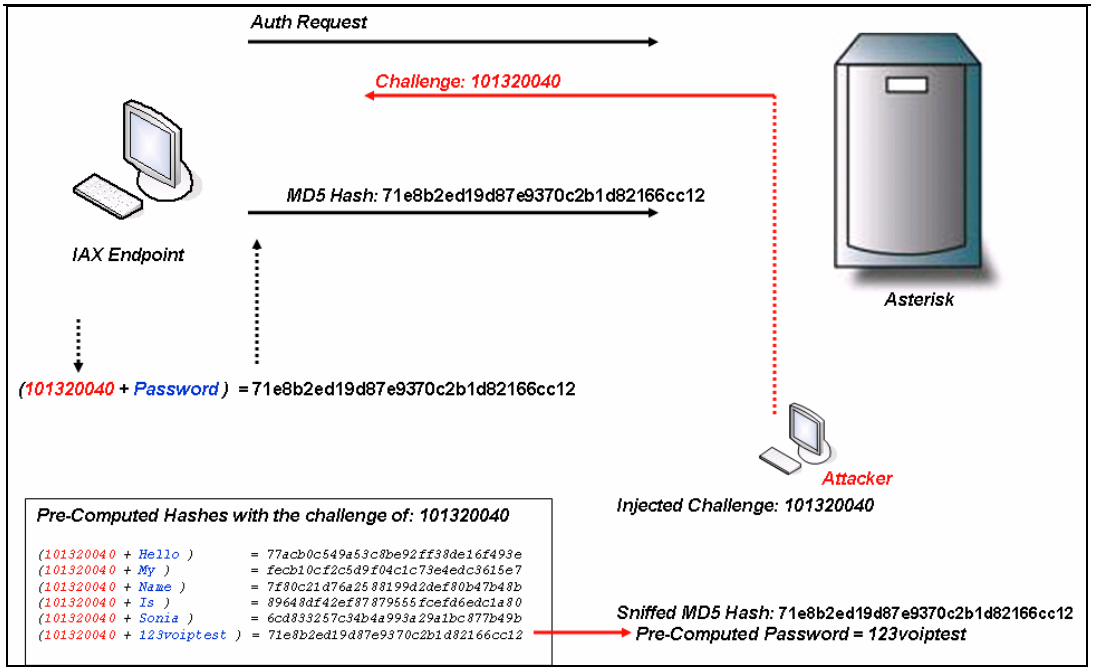
In addition to the passive attack, IAX is also vulnerable to a pre-computed dictionary attack. Unlike the passive dictionary attack where the attacker sniffs the challenge and MD5 hash over the network, the pre-computed attack would require the attacker to take a known challenge and concatenate it with a list of password to create a long list of MD5 hashes. Once a list of pre-computed hashes are created, the attacker would inject a challenge, the same challenge that was used to create the pre-computed hashes, to an IAX client endpoint. When the endpoint receives the challenge from the attacker, it will respond with an MD5 hash, which is derived from the attacker's challenge and the endpoint's password. Since the challenge was sent by the attacker, the attacker does not need to brute-force the MD5 hash, but rather simply match the hash sent by the endpoint to a pre-computed hash created by

[6] Further discussed in Section 3.1 of this paper

the attacker. Once the one of the hashes match, the attacker knows he or she has compromised the password.

The pre-computed dictionary attack significantly reduces the time required to gain an endpoint's password since the long list of MD5 hashes has already been created. Additionally, since the attacker is allowed to inject a challenge during the authentication process, by spoofing the source IP of the Asterisk server and predicting the sequence information of the IAX packet, the attacker is guaranteed that their challenge will be used by the endpoint in order to create the MD5 authentication hash. For example, if an attacker concatenates '101320040' with every word in the English dictionary, they have a list of pre-computed hashes. The only step the attacker would need to do is inject a packet to the endpoint with the challenge of '101320040'. When the endpoint receives the challenge, they will send the MD5 hash over the network. The attacker can simply sniff the response and compare it to the pre-computed list. Once one of the pre-computed MD5 hash matches the one capture from the target, the attacker knows the password. Figure 1.4 shows the example of the pre-computed attack using active packet injection.

Figure 1.4



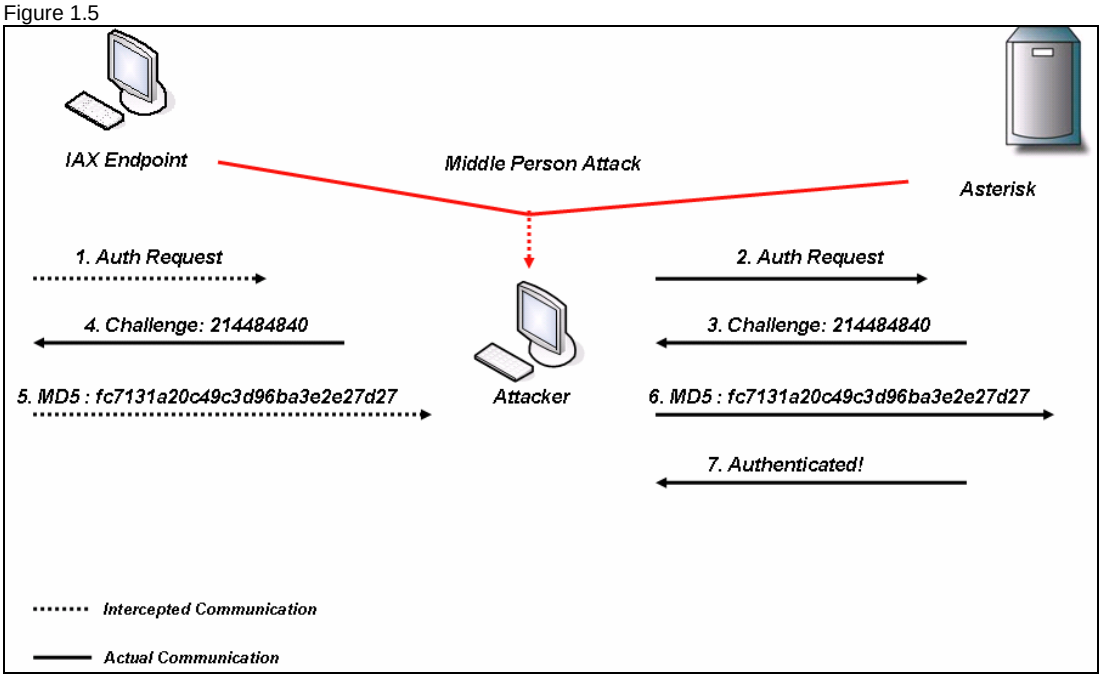
Notice in figure 1.4, the attacker has created a list of pre-computed hashes based on the challenge of 101320040 (shown on the lower left). The attacker then injects the challenge of 101320040 during the endpoint's authentication process. The client creates an MD5 hash from using the attacker's challenge. Unlike the passive dictionary attacker where the attacker needs to brute-force the password, once the attacker sniffs the MD5 hash over the network, they can simply match the sniffed MD5 hash to one of the pre-computed MD5 hashes. If a match appears, the attacker has just gain access to the endpoint's password.

1.1.3 Middle Person Attack[7]

In addition to active attacks, IAX's support of the challenge response authentication method makes it vulnerable to a middle person attack. The middle person attack would require an attacker to perform an attack on the network via ARP cache poisoning or DNS spoofing techniques. Once the attacker is routing traffic between a legitimate endpoint and the Asterisk server, he/she can perform a middle person attack and authenticate to the Asterisk server without knowing a valid username/password.

During an attack, the malicious user monitors the network to identify when an IAX endpoint sends an authentication request to the Asterisk server. When the authentication request occurs, the attacker would intercept the packets, since they are performing a middle person attack , and prevent it from reaching the real Asterisk server. The attacker would then send their own authentication request to the Asterisk server. Using the challenge response method for authentication, the Asterisk server will send a challenge to the attacker. The attacker receives the challenge and then sends the same challenge to the legitimate endpoint, who was wishing to authenticate in the first step. The legitimate endpoint would then send a valid MD5 hash to the attacker (derived from the real password and Asterisk's challenge), thinking the attacker is the actual Asterisk server. Once the attacker has the valid MD5 hash from the legitimate endpoint, the attacker would send the hash to the Asterisk server and become successfully authenticated. See figure 1.5 for details.

Figure 1.5



The middle person attack significantly increases the attack surface on IAX implementations, allowing an attacker become authenticated to the Asterisk server without or brute-forcing a single username/password.

---

[7] Middle Person attack is also know as a Man-in-the-Middle (MITM) attack

1.2.1 MD5 to Plaintext Downgrade Attack

The IAX protocol specification does not mention some important security protections, leaving implementations potentially vulnerable to active attacks. This susceptibility to active attacks arises from the fact that no integrity protection is provided in the IAX protocol. Due to the lack of integrity protection, both the endpoint and the Asterisk server can never be sure that their communication has not been tampered with on the wire.

Previously, IAX clients built with the IAX implementation library 'iaxclient' were vulnerable to an authentication downgrade attack. Vulnerable IAX implementations could be downgraded to plain-text during the authentication process. The downgrade attack caused an endpoint to send its password in clear-text, when it would normally use a MD5 Digest for authentication.

In order to perform this attack, the attacker needs to complete a few steps. First, the attacker would need to sniff the network[8], watching for an endpoint attempting to register to the Asterisk Server (AS) using a Registration Request (REGREQ) packet. The attacker would then need to parse out the required sequence values from the REGREQ packet, including the Destination Call ID (DCID), Outbound Sequence Number (oseq), Inbound Sequence Number (iseq), Username Length, and Username. Once the information has been gathered, the attacker needs to increase the iseq value to correspond with the existing session originally created by the AS (making it valid for a spoofed REGAUTH packet). After the sequence information is increased appropriately, the attacker would then inject a REGAUTH packet, which is spoofed from the AS to the endpoint, specifying that only plaintext authentication is allowed. If the spoofed packet wins the race back to the endpoint (ahead of the AS's real packet that requires MD5 authentication), the endpoint will send another REGREQ packet across the network with the password in plaintext. This allows the attacker to recover the password from the network with a standard sniffer such as Wireshark[9]. See figure 1.6 for an example.

[8] All exposures listed in the whitepaper require access to the network traffic between any victim endpoints and an Asterisk server. Gaining access to network traffic in a switched environment is outside the scope of this document, but is generally considered a trivial attack. For example, see papers such as http://www.sans.org/resources/idfaq/switched_network.php and tools such as arpspoof in the dsniff package http://www.monkey.org/~dugsong/dsniff/.

[9] http://www.wireshark.org

Figure 1.6

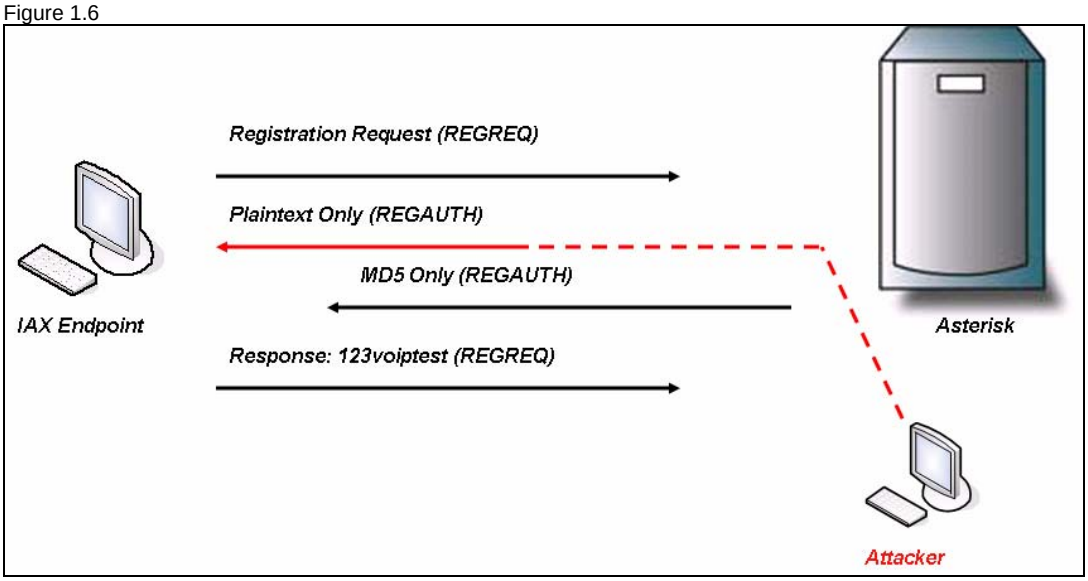


Figure 1.6 shows an endpoint attempting to register to the Asterisk server. During the authentication process, the attacker pulls the required session information from this packet. Once the information is extracted, the attacker will then inject a REGAUTH packet specifying only plaintext authentication is allowed, spoofed from the Asterisk server to the endpoint. When the endpoint receives the spoofed packet, it will respond with another REGREQ with the password in plaintext (in this case, the sample password "123voiptest" is shown.) Because this password is sent in plaintext, it can be easily sniffed by an attacker.

During the course of testing, iSEC contacted the iaxclient library developers and worked with them to resolve this issue. The iaxclient developers responded by removing plaintext authentication functionality entirely from the library. Since this issue has been resolved in the latest version of iaxclient, iSEC Partners has released an automated Python tool, called IAXAuthJack.py, to test if existing iaxclients are vulnerable to this attack. The tool is described further in Section 3.2

2.0 Denial of Service Attacks

A Denial of Service (DoS) attack can leave an endpoint unusable or unavailable for an extended period of time. The severity of DoS attacks are often different between one environment and the next. For example, a DoS attack on a NFS daemon may prevent end-users from gathering files over the network; however, a DoS attack on a VoIP network might prevent one user from calling 911 in case of an emergency. While any type of DoS attack is not desired, the severity of a DoS on VoIP networks can often be higher due to the dependencies of end users on voice communication.

Similar to the downgrade attack against authentication, the Denial of Service attacks against IAX are the result of predictable session information and a lack of integrity protection. This allows an active attacker to spoof control frames. While DoS issues are openly discussed in the IAX RFC, iSEC Partners wishes to demonstrate the ease in which some of the attacks can be performed. Additionally, it should be noted that even if AES encryption is used during a call, these attacks are still valid as all session information remains in the clear.

The following section discusses a few of the DoS attacks identified in the IAX protocol.

2.1 The Registration Reject (REGREJ) Attack

The Registration Reject attack is aimed at preventing an endpoint from registering to the Asterisk Server (AS). In this attack, an attacker monitors the network for a victim endpoint attempting to register to the AS via a Registration Request (REGREQ) packet. The attacker parses out the required values from the REGREQ packet, such as the Destination Call ID (DCID), Outbound Sequence Number (oseq), Inbound Sequence Number (iseq), Username Length, and Username. Once the information have been enumerated, the attacker needs to increase the iseq value to correspond with the existing session originally created by the AS (making it valid for a spoofed REGREJ packet). After the sequence information in increased appropriately, the attacker would then inject a REGREJ packet spoofed from the AS to the victim. This attack depends on the attacker's packet winning the race against the server's REGAUTH packet, which would allow the registration process to continue. See figure 2.1 for an example.

Figure 2.1

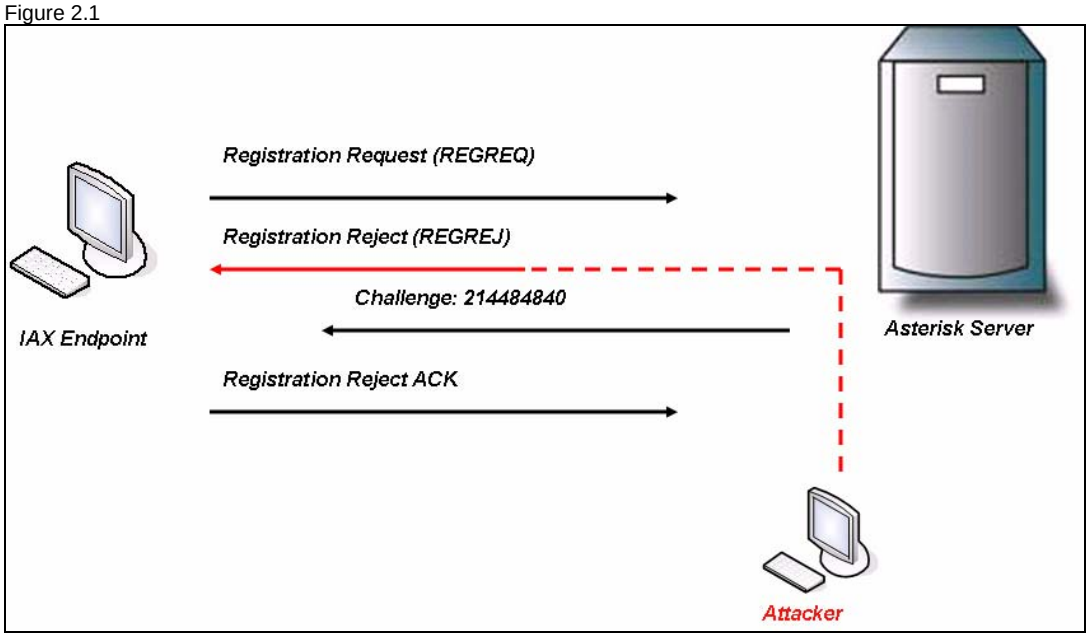


Figure 2.1 shows an endpoint attempting to register to an Asterisk server. During the authentication process, the attacker then pulls the required session information from this packet. Once the information is enumerated, the attacker will then inject a REGREJ packet, specifying the authentication process has failed. When the endpoint receives the spoofed packet, it will believe registration process has failed and ignore the server's MD5 challenge.

2.2 The HangUP Attack

The HangUP attack disconnects calls that are in progress between two endpoints. During the attack, a malicious endpoint monitors the network for the many items that indicate a call is in progress, such as an ANSWER packet, a Mini voice packet, a PING or PONG packet. The attacker then parses out required values from one of these packets, such as the Source Call ID (SCID), Destination Call ID (DCID), Inbound Sequence Number (iseq), and Outbound Sequence Number (oseq). Once the information has been parsed, the attacker must manipulates the iseq and oseq values so they will be valid for a spoofed HANGUP packet, ensuring their sequence information increases correctly to match the values sniffed over the network. Finally, the attacker injects the spoofed HANGUP packet with the correct information, which will cause a call to be torn down. See figure 2.2 for an example.

Figure 2.2

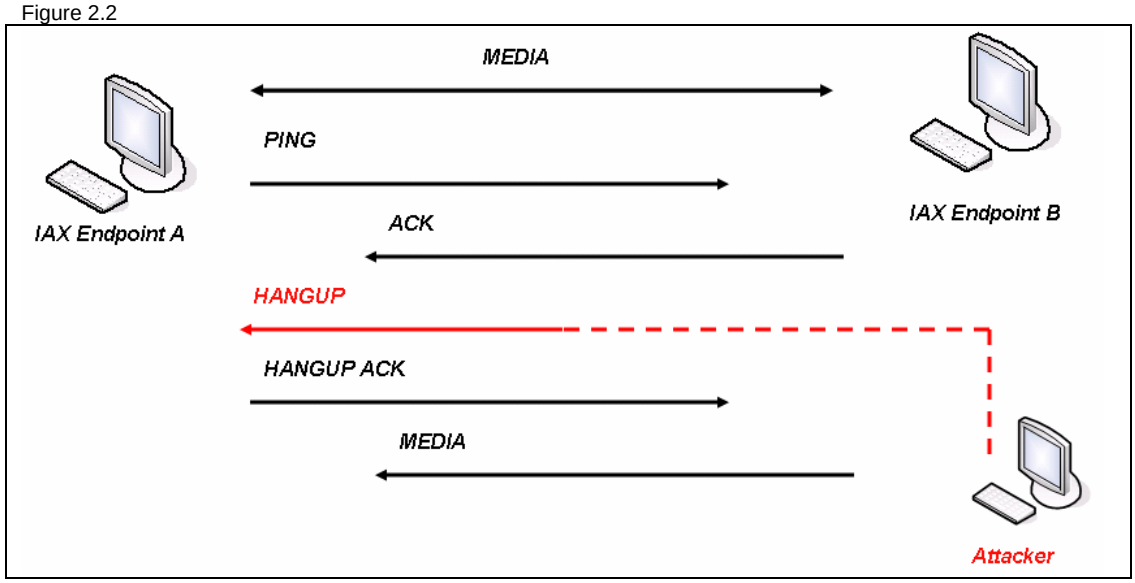


Figure 2.2 shows an existing call between to endpoints, with media flowing in both directions. During a phone call, a control frame is sent across the network (a PING in our example), which contains session information an attacker needs to complete this attack. From the session information sniffed by the attacker, a spoof HANGUP packet is created and sent to endpoint A. Once Endpoint A receives the information, the existing phone call will be torn down. From this point on, Endpoint A will no longer continue the call even though Endpoint B continues sending media as it is unaware the HANGUP has taken place.

iSEC has created IAXHangup.py, which is a Python-based tool to automate this attack. The tool is described further in Section 3.3.

2.3 The Hold (QUELCH) Attack

The Hold (QUELCH) attack is aimed at disrupting calls which are in progress between two endpoints. The attack occurs by one placing one side on 'Hold' while not notifying the other side. In this attack, a malicious endpoint monitors the network for the many signs indicating a call is in progress, such as an ANSWER packet, a Mini voice packet, a PING or PONG packet. The attacker parses out required values from one of the packets, such as the Source Call ID (SCID), Destination Call ID (DCID), Inbound Sequence Number (iseq), and Outbound Sequence Number (oseq). Once the information has been parsed, the attacker must manipulates the iseq and oseq values so they will be valid for a spoofed Hold (QUELCH) packet, ensuring their sequence information increases correctly to match the values sniffed over the network. Finally, the attacker injects the spoofed QUELCH packet, which will cause one side of the conversation to unknowingly be placed on hold See figure 2.3 for an example.

Figure 2.3

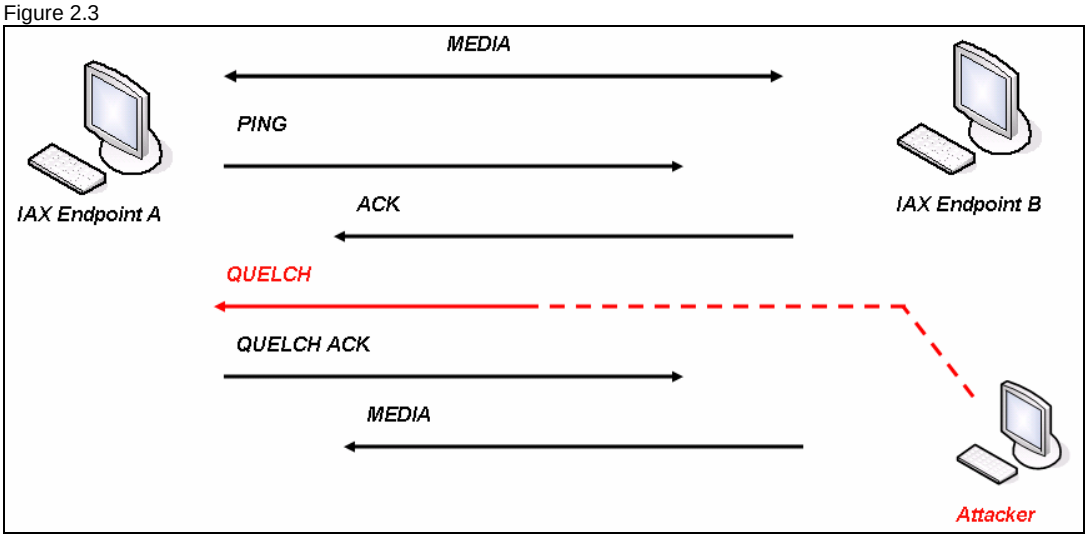


Figure 2.3 shows an existing call between two endpoints, with media flowing in both directions. During a phone call, a control frame is sent across the network (a PING in our example), which contains session information an attacker needs to complete this attack. From the session information sniffed by the attacker, a spoof QUELCH packet is created and sent to endpoint A. From this point on the connection becomes one-sided (but not disconnected) as Endpoint A will no longer send media to Endpoint B.

iSEC has created IAXHangup.py, which is a Python-based tool that can be easily modified to automate this attack. The tool is described further in Section 3.3.

2.4 The Call Reject (REJECT) Attack

The Call Reject attack is aimed at preventing calls from being accepted. In this attack, a malicious endpoint monitors the network for packets indicating that a call is being setup, such as a NEW, ACCEPT, or RINGING packet. The attacker parses out required values from one of these packets, such as the Source Call ID (SCID), Destination Call ID (DCID), Inbound Sequence Number (iseq), and Outbound Sequence Number (oseq). Once the information has been parsed, the attacker must manipulates the iseq and oseq values so they will be valid for a spoofed REJECT packet, ensuring their sequence information increases correctly to match the values sniffed over the network. The attacker then creates a spoofed packet with these values, the source IP and MAC address of the callee, and the destination IP and MAC address of the caller. Finally, the attacker sends the spoofed REJECT packet to the caller, causing the caller to believe that the call has been rejected by the callee. This attack depends on the attacker's packet winning a race against the callee's ANSWER packet, allowing the call to be established. See figure 2.4 for an example.

Figure 2.4

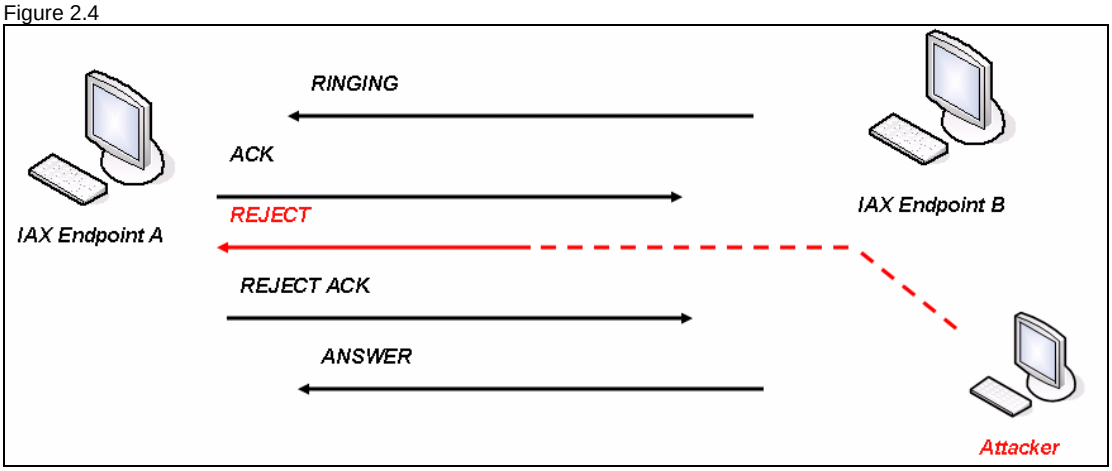


Figure 2.4 shows an attacker monitoring the network for a call setup packet, in this case RINGING, which indicates that an endpoint is attempting to place a call. The attacker then pulls the required session information from this packet to construct and inject a spoofed REJECT packet. When the endpoint receives this packet it believes the call has been rejected and ignores any further control packets for this call.

3.0 VoIP Security Tools for IAX

The following sections describe iSEC's IAX security tools for VoIP. The tools that are include the following list:

- IAX Brute
- IAXAuthJack
- IAXHangup

3.1 Password Grinder – IAX.Brute

IAX.Brute is a passive dictionary attack tool on the challenge/response authentication method supported in VoIP IAX implementations. The program is written primarily to test VoIP networks that use IAX for voice communication. The proof of concept tool shows how the challenge/response authentication process used by IAX endpoints is vulnerable to an offline brute-force attack. This attack allows malicious users to steal passwords and hijack endpoint identities.

IAX.Brute requires the user to sniff the challenge and the MD5 hash between two IAX endpoints. Both the challenge and MD5 hash are sent over the network in clear-text, making the process an easy task. Once the user has captured both items, IAX.Brute will grind the endpoint's password using any dictionary file supplied by the user (IAX.Brute includes a standard dictionary file with over 280,000 thousand common passwords). IAX.Brute will then create an MD5 hash from the user supplied challenge and a word in the dictionary file. Once the MD5 hash generated by the tool matches the MD5 hash sniffed over the network, the user has then compromised the IAX endpoint's password using the IAX.Brute tool. See items 3.1 thru 3.3 as an example:

Figure 3.1 show the challenge (214484840) and username (voiptest1) sniffed over the network in clear text:

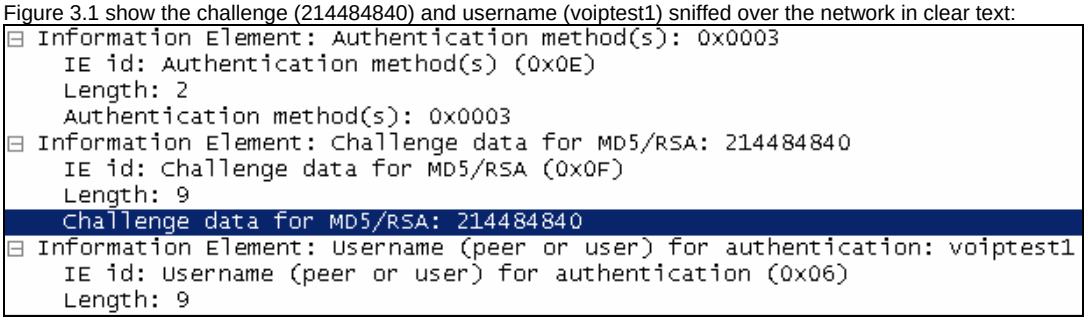


Figure 3.2 show the MD5 hash sniffed over the network in clear text:

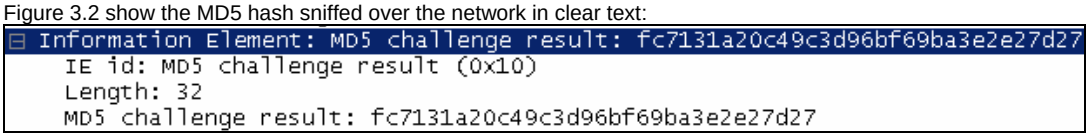
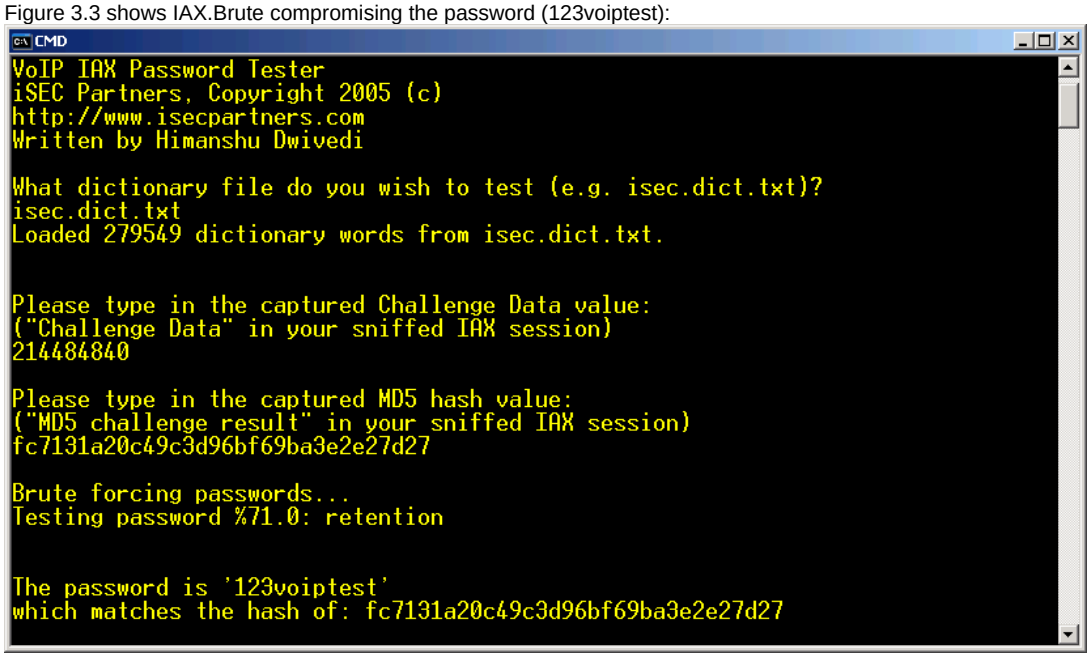


Figure 3.3 shows IAX.Brute compromising the password (123voiptest):



Notice in figure 3.3, IAX.Brute simply walks through three steps to identify the password:

1. IAX.Brute supplies dictionary file
    a. isec.dict.txt is included with the tool, but any dictionary file can be used
2. User supplies challenge
    a. User enters challenge sniffed over the network, which is 214484840 from figure 4.1
3. User supplies MD5 hash
    a. User enters MD5 hash sniffed over the network, which his fc7131a20c49c3d96bf69ba3e2e27d27 from figure 4.2
4. IAX.Brute performs the passive dictionary attack
    a. Done! IAX.Brute performs the passive dictionary and identifies the password, which is “123voiptest”!

3.2 Authentication Downgrade – IAXAuthJack.py

IAXAuthJack.py is a tool used to actively perform an authentication downgrade attack and force an endpoint to reveal its password in plaintext over the network. It performs this attack by sniffing the network for traffic indicating that a registration is taking place, and then injecting a REGAUTH specifying that the endpoint should authenticate in plaintext rather then MD5 or RSA. The tool has two modes of operation, which are described below:

Targeted attack:

```
iaxauthjack.py -i eth0 -c EndpointIP -s ServerIP
```

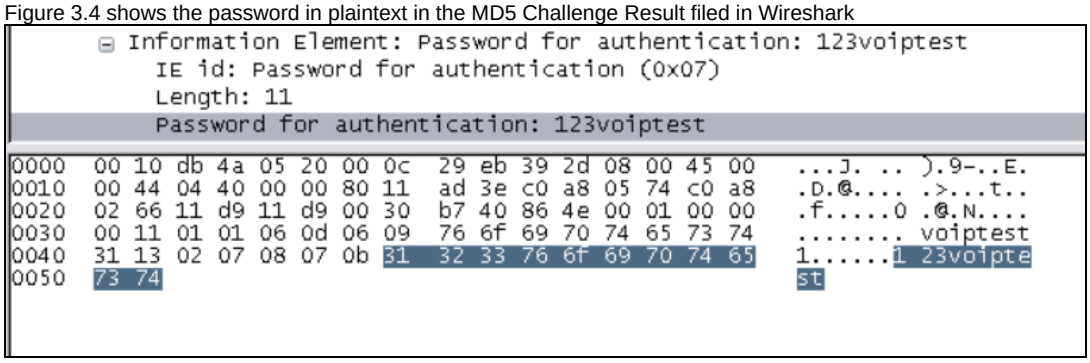
In this example, the tool will listen on the “eth0” Ethernet interface for control frames from a specific IAX endpoint, whose IP address is specified by the “-c” argument, who is attempting to register to the server, whose IP address is specified by the “-s” argument. IAXAuthJack.py will then inject a REGAUTH packet spoofed from the server to the endpoint, causing the endpoint to respond with a REGREQ packet with the password in plaintext.

Wildcard attack:

```
iaxauthjack.py -i eth0 -a -s ServerIP
```

In this example, the tool will listen on the "eth0" Ethernet interface for control frames from ANY IAX endpoint who is attempting to register to the server, whose IP address is specified by the "-s" argument. IAXAuthJack.py will then inject a REGAUTH packet spoofed from the server to the endpoint, causing the endpoint to respond with a REGREQ packet with the password in plaintext.

Figure 3.4 shows the password in plaintext in the MD5 Challenge Result filed in Wireshark



3.3 Call Hangup – IAXHangup.py

The IAXHangup.py tool is used to disconnect IAX calls. It first monitors the network in order to determine if a call is taking place. Once a call has been identified, it then injects a HANGUP control frame into the call. The tool has two modes of operation, which are described below:

Targeted attack:

```
iaxhangup.py -i eth0 -a 1.1.1.1 -b 2.2.2.2
```

In this example, the tool will listen on the "eth0" Ethernet interface for control frames indicating that a call is taking place between hosts 1.1.1.1 and 2.2.2.2. IAXHangup.py will then inject a HANGUP and disconnect the call.

Wildcard attack:

```
iaxhangup.py -i eth0 -e
```

In this example, the tool will listen on the "eth0" Ethernet interface for control frames indicating that a call is taking place between any hosts on the network. IAXHangup.py will then inject a HANGUP and disconnect the call.

Conclusion and Recommendation

IAX has the potential to be a very popular protocol for VoIP architectures. Key IAX features can all make it a popular choice in VoIP hard and soft phones, including the simplistic nature of the protocol, the use of a single UDP port, the use of one protocol for signaling and media transfer, the use of a few network components (IAX clients and servers only versus multiple media proxies, gateways, gatekeepers, STUN servers) and its friendliness to network firewalls. While IAX has many operational and functional advantages over SIP or H.323, it may not contain a better security story when compared with authentication, encryption, and availability. The authentication weaknesses of SIP and H.323 are mirrored in IAX. Additionally, the lack of use and/or support for encryption with RTP media transfers is similar to the IAX in terms of voice communication. Furthermore, IAX's susceptibility to denial of service attacks are similar to SIP and H.323. These facts leave IAX in the same boat with SIP, H.323, and RTP in terms of security at present.

The possible security benefits of IAX, as listed in its RFC, can be achieved once the support for proper authentication and encryption appear on IAX endpoints and servers. For example, IAX can support RSA public and private key pair for authentication, allowing for a stronger authentication model from passive and active network attacks (it should be noted that the deployment challenges with public and private keys with every VoIP phone will be difficult). Additionally, AES encryption with a pre-set shared secret can encrypt media communication, preventing passive attackers from eavesdropping telephone conversation (assuming the pre-set shared secret is not vulnerable to an active or passive dictionary attack during call setup). However, while eavesdropping will be prevented in a properly encrypted call, IAX will still be susceptible to Denial of Service attacks due to session information remaining in the clear.

In addition to support for RSA authentication and AES call encryption, iSEC Partners recommends that the language in future versions of the IAX RFC be changed to note that use of MD5 authentication should not be considered secure. Finally, IAX implementations will need to continue to guard against implementation flaws such as authentication downgrade attacks. Upon the arrival of these features, along with a proper way to implement them, the IAX security story can be drastically better than current VoIP protocols used on the market.