VOIP (Voice Over Internet Protocol) Hacking-Fake Calling

Author: Avinash Singh
Co-Author: Akash Shukla

Avinash Singh
Corporate Trainer
(Virscent Technologies Pvt. Ltd.)
Appin Certified Ethical Hacker (ACEH)

Email: avinash.singh@virscent.com
avinash546@gmail.com

SNL: https://facebook.com/avinash546

Biography of Author:

Avinash Singh is an IT Security Researcher and Evangelist currently working with **Virscent Technologies Pvt. Ltd.** as a Research & Development Analyst CUM Corporate Trainer, **Appin Certified Ethical Hacker (ACEH)** and done his **B.tech** from **CSE branch**. He has also worked with **Kyrion Technologies Pvt. Ltd.** for 8 months as a Research and Development Analyst CUM Corporate Trainer.

His expertise includes **PC Hardware and Networking**, **LINUX, Core Java, exploit research**. He has also done training in **MCSE, LINUX, CCNA** and basics of **Firewalls**. He has done lot of research work on many projects in which some of them are:
a) IDS: Snort & OSSIM
b) Firewall: Pfsense
c) Honeypot Technology (Honeyd)
d) **Research and working of VOIP (Voice Over Internet Protocol)**

He has trained more than **2000+ students** and having more than **5 years'** experience of IT Security field. He has conducted lot of workshops around the nation. He is better known for his work on **VOIP hacking.**

Akash Shukla
Corporate Trainer
(Virscent Technologies Pvt. Ltd.)
Certified Information Security Expert (CISE)

Web: www.hackersreloaded.com

Email: akash@hackersreloaded.com
akash@virscent.com
aks.zooming@gmail.com

SNL: https://facebook.com/hybridakash
https://twitter.com/akszooming

Biography of CO-Author:

Akash Shukla is an IT Security researcher currently working with **Virscent Technologies Pvt. Ltd**. as a **corporate Trainer**. He is a **Certified Information Security Expert (CISE)**, **Founder and Admin of Hackersreloaded.com**, **Member of Computer Society of India (CSI)** as well as worked with **Kyrion Technologies Pvt. Ltd.** as a **Penetration Tester** for Two months. His expertise includes Research and Development in this domain, Computer and Network Security, exploit research, C, PHP, Perl, Core JAVA and website designing.

He has trained more than **1000+ students** and having more than 4 years' experience of IT Security field. He has conducted lots of workshops around the nation. He has also found some big security loopholes in high rank websites of cyber space like **Facebook, Orkut, Ebay, Paisalive**.

He has also made possible calling from OLD version of **Micromax modem MMX310G** whose process video got 1000's of hits on youtube. He has some videos on youtube because he strictly share only new hack stuffs ☺.He is also running his own free-lance organization **RMAR Dizi Securities (Venture of Cyber Security Experts)**.

He has also helped in curing the vulnerability of **Gautam Budhh Technical University's** website which was earlier hacked by him and reported to the Vice Chancellor as well, after which Vice Chancellor admired him for his sincere efforts in reporting the vulnerability, the news was also in Media. He has also worked on many projects in which **Snort-An Open Source Intrusion Detection System** was the project whose LOG problem on Windows 7 was resolved by him and successfully patched. Earlier he worked with various companies as a free-lancer.

Abstract:

What Is AsteriskNOW?

AsteriskNOW is an open source software appliance, a customized Linux distribution that includes Asterisk, the Asterisk GUI, and all other software needed for an Asterisk system. The Asterisk GUI gives you the ability to easily configure your Asterisk system without being a technical expert.

Before You Begin

AsteriskNOW installation is easy, because the appliance includes only those components necessary to run, debug, and build Asterisk. You no longer have to worry about kernel versions and package dependencies. AsteriskNOW is a custom Linux distribution for Asterisk based on rPath Linux.

What You Will Need

- A system on which you can install AsteriskNOW
- A CD writer and associated software
- Connection to the Internet
- Firefox browser

The Asterisk GUI currently requires the Firefox browser (available at http://www.mozilla.com/en-US/ for optimum performance. Wider browser support will be available with future versions.

Installation

You should observe all normal precautions when preparing and installing a new distribution. Any existing operating systems on your hard drive will be removed by the Express Installation

Accessing the GUI

Once you have completed your installation and rebooted your machine, you will be able to access the Asterisk GUI. The URL used to access the Asterisk GUI is the IP address or hostname displayed after rebooting your machine. Enter this IP address in your browser URL. You will be able to refine your AsteriskNOW installation by accessing the Asterisk GUI.
The default username is admin and password is password.

Making and Receiving Calls

Specifically, we'll:

- Register an account with a VoIP (voice over IP) provider.
- Configure Asterisk to connect to our VoIP provider.
- Create a SIP extension to make and receive calls from.
- Configure Asterisk rules for making calls.
- Configure Asterisk rules for receiving calls.
- Connect a softphone (software phone) to our Asterisk server as an extension and use it to make real calls.

So let's get started!

Getting a VoIP Account

The first thing we need to do is sign up with a VoIP (voice over IP) provider.

VoIP (voice over IP) is a relatively new way to connect to the PSTN using the Internet. VoIP is extremely low-cost compared to other methods, and very easy to set up as it only requires an active Internet connection to use. More and more businesses are switching over to VoIP for cost reasons.

There are two main types of VoIP protocols that people use with Asterisk. There is SIP (session initiation protocol) and IAX (inter-asterisk exchange). While both of these protocols are still widely in use, SIP has become the dominant VoIP protocol in today's world. In fact, the term SIP has mostly replaced VoIP. When speaking to people in the telephony industry, it is common to hear people say 'I have so and so as my SIP provider'. So from now on, we'll use that terminology as well.

Picking a SIP provider is a very important task. If you Google for 'SIP providers' you'll find thousands of options. Here are few of them:

- Flowroute
- voip.ms
- Vitelity

- Bandwidth
- Voicepulse

Flowroute is my favorite SIP provider because they have a great website, low prices, and business-class reliability. So instead of walking through the set up for each of the above providers (that would take forever) I'll just cover Flowroute. If you choose to go with another provider, you can still follow along with this article and get a sense of what to do.

Create an Account

To create an account with Flowroute, visit their sign up page and fill in your information. You don't need a credit card, and they'll give you 25 cents of credit (enough to make approximately 30 minutes of calls). If you want to be able to receive calls as well as make them, then you'll have to deposit some money into your account as you'll need to own a DID (more commonly known as phone number) which costs a bit more than 25 cents.

Once your account has been created, you'll be directed to the account dashboard, where, if you want, you can deposit some money via Amazon Payments.

Create a SIP Extension

Now that we've set up our SIP trunk, it's time to create an extension. An extension is just another term for 'phone'. SIP extensions (IP phones) are the most common type of extensions in use on modern phone systems. Sure, you can still hook up those old analog phones to your Asterisk PBX, but we'll save that for another day. Today we'll create a simple SIP extension which we will later hook up a soft phone to and use to make and receive calls.

To define a SIP extension, we need to pick several things:

- **An extension number.** This is typically a numeric number, several digits in length. I'll be using 1000 in the following examples.
- **A secret password.** This is generally an alphanumeric password of any length used to 'secure' your extension. Any device which wants to make or receive calls from your extension need to know this password in order to authenticate. This password does not need to be remembered, and should be globally unique.

Creating the Extension

Now that you've picked your extension number and secret, let's create it!

Scroll down to the very bottom of your /etc/asterisk/sip.conf file, (below the SIP trunk we created in the previous section), and insert the following (make sure you swap out my extension number for yours, and my secret for yours):

```
[1000]
type=friend
nat=yes
canreinvite=no
```

```
secret=mysecretpassword
qualify=yes
mailbox=1000@default
host=dynamic
dtmfmode=rfc2833
dial=SIP/1000
context=outgoing
```

At this point we've fully configured a Flowroute SIP trunk and a SIP extension with Asterisk. The next thing for us to do will be configuring our network so that our SIP trunk works.

Configure the Dial Plan

Since we have our SIP trunk working, we now need to program Asterisk, and teach it how to route calls for us. There are two things that we need to teach Asterisk to do: how to route outgoing calls, and how to route incoming calls. Each of these requires separate Asterisk configuration to work.

Our goal will be to write the following two rules:

1. When we dial an 11-digit US phone number from our soft phone, it should call that phone number.
2. When someone calls our SIP DID (phone number), we should connect that call to our soft phone so we can talk to the person who called us.

There are several ways to program Asterisk to handle call routing, but the simplest (and native) way to do it is via 'dial plan'. Dial plan is the name of Asterisk's own scripting language. It is very simple, and easy to understand (even for non-programmers).

In the next few sections, we'll write dial plan code to route calls according to our rules above.

Start With a Clean Slate

Before we begin writing our code, let's quickly create a nice clean extensions.conf file to work in. Remove your current dial plan file (/etc/asterisk/extensions.conf) and replace it with the following:

```
[general]
static=yes
writeprotect=no
clearglobalvars=no

[globals]
TRUNK=flowroute
```

All I did here was remove all comments and clutter from the file, so that we can clearly see what we're doing. I also added a global variable called TRUNK which contains the name of our SIP trunk defined earlier in the article. We'll use this global variable later on to make outbound calls.

Configure Outbound Routing

The goal here is to allow our extension to dial an 11-digit US phone number, and connect it to the PSTN via our Flowroute SIP trunk.

Remember when we configured our SIP extension? One of the keys we specified in the configuration file was: context=outgoing. The context field of an extension determines what dial plan context the pattern matching will start at. This is similar to the main() function in most programming languages.

The context specified in our SIP extension plays an important role in routing outbound calls. It says (in human English):

When any extension who's 'context' key is equal to 'outgoing' dials a number, send the number that was dialed to the [outgoing] context to be processed. Let the [outgoing] context determine what to do at this point.

So now that we know what context we need to create (the [outgoing] context), let's make it!
Open your extensions.conf file and add the following code to it at the bottom:

```
[outgoing]
exten => _1NXXNXXXXXX,1,Dial(SIP/${EXTEN}@${GLOBAL(TRUNK)})
```

Let's go through the code and analyze exactly what is happening.

The first bit: `exten =>` is standard. All Asterisk dial plan code starts with this bit. The extension: `_1NXXNXXXXXX` is a regular expression that Asterisk will use to pattern match the number dialed. Remember above how we said that we're going to route all 11-digit US telephone numbers outbound? This pattern represents an 11-digit US telephone number. Since all 11-digit US telephone numbers begin with the number 1, we hard-code that number. The variable N represents a number (2 through 9), and the variable X represents a number (0 through 9). Therefore, the pattern _1NXXNXXXXXX will match any 11-digit US telephone number.

After the extension (pattern), you'll see a comma character followed by the number 1. The number 1 is the priority. Asterisk dial plan code can contain more than a single line of instructions. In cases where there are multiple lines of code that need to be executed, Asterisk relies on the priority number to determine what to do first.

Imagine that we had code which looked like:

```
exten => _1NXXNXXXXXX,2,Dial(18002223333@flowroute)
exten => _1NXXNXXXXXX,1,Dial(19999999999@flowroute)
```

In this case, Asterisk would execute the code at priority number 1 first, then the code at priority number 2. So always check for priority numbers when looking at code as they will help determine what is going on.

The last part of the code you see is the called application: `Dial(SIP/${EXTEN}@${GLOBAL(TRUNK)})`. The application we're using here is the `Dial()` command. This application instructs Asterisk to dial the phone number out of our Flowroute SIP trunk and connect the call to our extension (eg: make an outbound call)!

Everything inside of the `${}` characters is a variable reference. Asterisk has several pre-defined 'channel variables' which are always accessible. In our `Dial()` code, we reference the `${EXTEN}` channel variable, which contains the number that was dialed and pattern matched. If we dialed the number 18002223333, then `${EXTEN}` would expand to 18002223333.

The `${GLOBAL(TRUNK)}` code references the global variable `TRUNK` which we defined at the top of our extensions.conf file. This code will expand to flowroute before executing the `Dial()` application.

So after all of the variable expansion is finished, Asterisk actually sees the following (assuming that we dialed the number 18002223333): `Dial(SIP/18002223333@flowroute)`, which is a lot easier to understand! The first part of the line that says SIP/ tells Asterisk that the number we're going to dial should be sent out of our SIP trunk. The @flowroute part tells Asterisk to dial the number 18002223333 on the flowroute SIP trunk, specifically. Doing it this way gives us flexibility. Imagine if we had many SIP trunks on our Asterisk system, and wanted to route certain calls through certain SIP trunks.

Outbound Routing, a Full Walk Through

Let's quickly perform a full walk through of what happens when we dial the number 18002223333 on our extension. (Don't worry that we haven't set up the soft phone yet, we'll get to that later.)

1. Asterisk receives a SIP call from our extension (1000) which has dialed the number 18002223333.
2. Asterisk looks at the context key in our extension's configuration settings, and sees that it is set to outgoing.
3. Asterisk sends the number 18002223333 to the `[outgoing]` context, defined in /etc/asterisk/extensions.conf to be pattern matched.
4. Once Asterisk finds a match for the number, it begins looking for the first application to execute by looking for priority 1.
5. Asterisk finds priority 1, and then beings performing variable expansion in the application field to prepare for execution.
6. Asterisk finishes variable expansion, then executes: `Dial(SIP/18002223333@flowroute)`.
7. Asterisk then calls the number 18002223333 on the flowroute SIP trunk, and connects the call to our extension so that both ends can talk to each other.

That's it for outbound routing!

Configure Inbound Routing

In order to route calls inbound, you will need a DID (phone number) with your SIP provider. This way, you can call your number, and it will direct to your Asterisk PBX system. A phone number is a lot like an IP address, each one is unique, and routes to a specific location.

Unfortunately, DIDs cost money (just like you pay for cell phone service and house phone service, you need to pay for VoIP service to rent a DID). If you would like to try out the code that follows, you'll need

to deposit some money into your Flowroute account, and then purchase a DID from the web panel. At this point in time, a single DID from Flowroute costs approximately $1.39 per month.
Once you've purchased a DID, write the number down somewhere. It will be an 11-digit phone number. For the code that follows, simply substitute in your DID where necessary, as I will be using the ficticious DID 18182223333.

Open your /etc/asterisk/extensions.conf file and add a new context called `[inbound]` to the bottom of your file (it can go beneath the `[outgoing]` context we created in the last section). The context should look like:

```
[inbound]
exten => 18182223333,1,Dial(SIP/1000)
```

The code here should look familiar to the code in the previous section. All we're doing is dialing the SIP extension 1000 (which is the extension we created earlier). You've probably noticed that there is no @trunk syntax at the end of our `Dial()` application. If you look back at your SIP configuration file (/etc/asterisk/sip.conf) you'll notice that when we created our extension, one of the keys we defined was: `dial=SIP/1000`, which tells Asterisk that in order to connect a call to that extension, we have to dial `SIP/1000`.

Inbound Routing, a Full Walk Through

Let's quickly perform a full walk through of what happens when someone calls our DID (18182223333) from the PSTN.

1. First our SIP provider will receive the call, and check to see what customer the DID belongs to.
2. The SIP provider will then check to see if we are registered to their SIP server.
3. Once they've confirmed our registration, they'll forward the call to our Asterisk system.
4. Asterisk will receive the call from our Flowroute SIP trunk, and check to see what context it should use to route the call.
5. Asterisk will see that our `[flowroute]` trunk is configured to use the context 'inbound', and will send the DID number, 18182223333 to the `[inbound]` context for pattern matching.
6. Asterisk will match the pattern 18182223333 in our `[inbound]` context, and begin looking for priority 1.
7. Asterisk will find priority 1, and then execute the `Dial(SIP/1000)` command.
8. Asterisk will connect the incoming call to our extension, so that both parties can talk.

Not too bad! We've set up the ability to receive incoming calls in only a single line of code!

At this point, we've created dial plan rules for both the outgoing and incoming call routing. Save your extensions.conf file move on.

Apply Your New Dial Plan Rules

Now that we've finished writing our dial plan code, we need to reload Asterisk to have it re-scan our extensions.conf file. To do this, simply type: asterisk -rx 'dialplan reload' from the command line.

The next section which will teach you how to hook up a soft phone to your Asterisk system so you can actually test your system!

Set Up a Softphone

In this section, we'll set up a soft phone to use to make calls. Soft phones are just SIP clients that can connect to Asterisk and act as normal phones. Asterisk can work with normal analog telephones as well as fancier (and more expensive) SIP phones, but SIP phones are much easier to set up, so we'll be configuring a soft phone today. If you want to hook up your analog phone to your Asterisk server–don't despair–that will be covered in another article in this series.

There are tons of soft phones to choose from, but I'll be walking you through using X-Lite, as it is one of the most popular and widely used.

First thing you'll want to do is download and install **X-Lite** on your computer (it runs on all platforms).

Once you've got it installed, open it up. If you don't have your extension information in front of you, open up your SIP configuration file (/etc/asterisk/sip.conf) and look at your extension definition.

Right click on the main window display in X-Lite and click on 'SIP Account Settings'. Now click 'Add…' to add a SIP account.

Here are the values you should fill in:

- **Display Name**: Set this to anything you like. I prefer 1000 (the extension I'm using).
- **User Name**: Set this to your extension number (1000).
- **Password**: Set this to your secret.
- **Authorization User Name**: Set this to your extension number (1000).
- **Domain**: Set this to the IP address of your Asterisk server.
- Make sure the box next to 'Register with domain and receive incoming calls' is checked.

Once you've configured all those settings, press OK and then close out of the configuration menu.

If your network and SIP configuration files have been properly configured, your X-Lite phone will now say 'Registered' at the top of the window! If it is not working, read back through the setup instructions and make sure you didn't miss anything. If you are using a virtual machine, also make sure your host network settings have been configured to receive incoming traffic.

Making and Receiving Your First Call

Now that we've gotten everything set up, let's actually make some calls!

On your X-Lite soft phone, go ahead and dial any 11-digit US telephone number (since this is the rule that we configured for outbound routing). When you hit dial (the green button) you'll make a call just like you would on a normal phone, and you'll be connected to the phone number you dialed!

If you want to receive a call, use your cell phone to call your DID, and Asterisk will route the call directly to your X-Lite phone. You will see and hear you X-Lite phone ring, and you can pick up the call by clicking the green button.

Conclusion

This was a very large article, and took a considerable amount of time to write. I hope that if you got this far, you were able to clearly configure and understand the basic Asterisk setup required to make outbound calls, and to receive inbound calls.

The material covered in this part of the series is bulky, but very important in understanding the way Asterisk works. In future parts of the Transparent Telephony series, we’ll have significantly shorter, more targeted articles, so following along should be easier.

Call Spoofing

Change the extension number that was given while creating the extension (Refer Page 9).
The call will go with the number specified by you.

Thanks ☺