

# Casos de uso

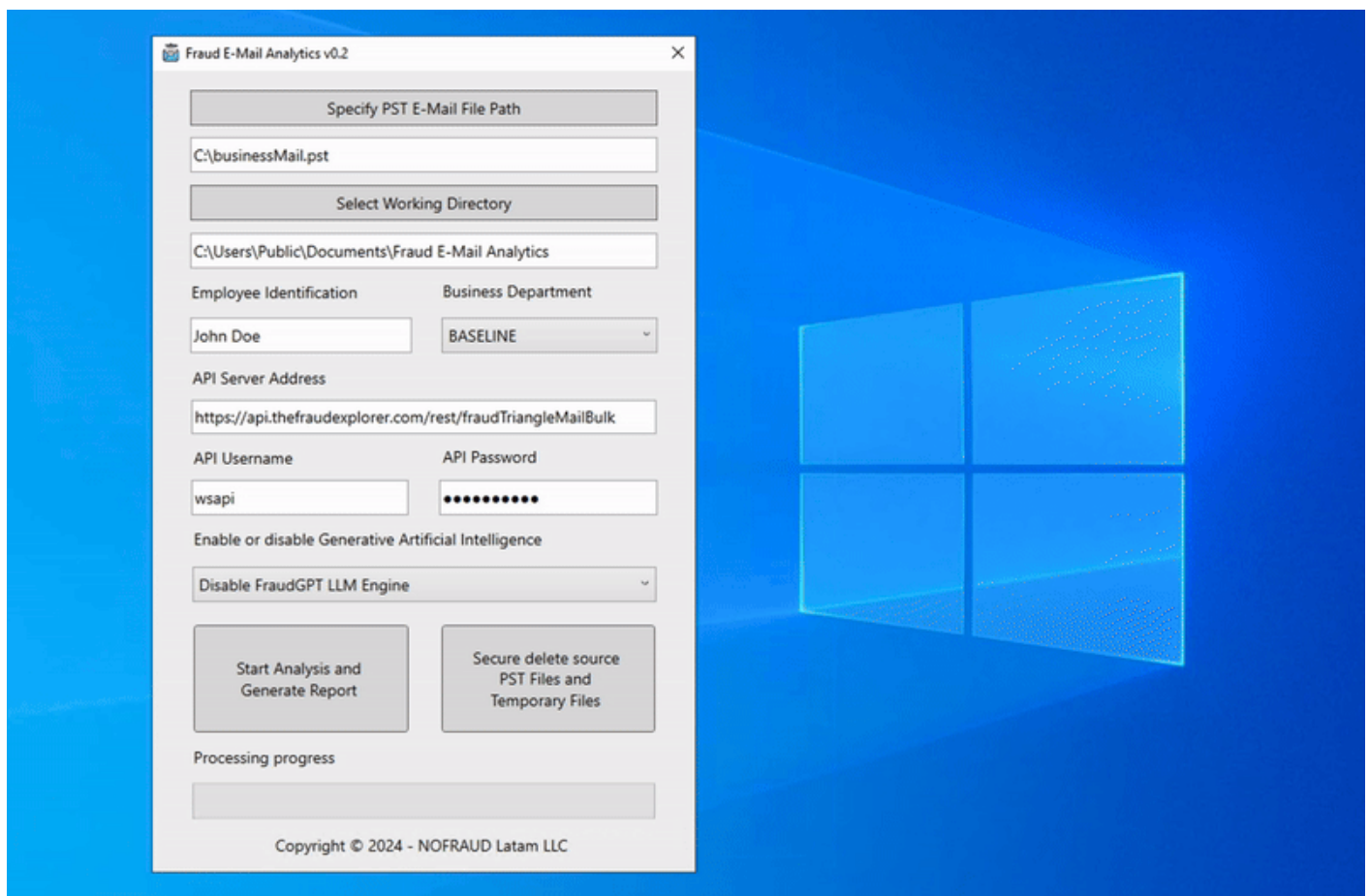
Diferentes implementaciones de API para casos cotidianos donde se quiere prevenir y detectar el fraude y la corrupción.

- Análisis de correos PST
- Análisis de la voz en un IPPBX

# Análisis de correos PST

**Fraud E-Mail Analytics** es un desarrollo de NOFRAUD que hace uso de la API **fraudTriangleMailBulk** con el objetivo de analizar los correos y chats alojados en un archivo PST con el algoritmo del triángulo del fraude y FraudGPT.

Una vez abierto, se selecciona el archivo PST, se especifica a qué persona pertenece el PST, el departamento al que fue asignado y al final se especifica si se quiere usar la IA Generativa Fraud GPT o no. Los datos del acceso a la API se cargan automáticamente y se componen de una URL, un usuario y password.



Una vez se tienen estos datos correctamente especificados, se procede a dar clic en **Start Analysis and Generate Report** y el software en ese momento realiza internamente las siguientes acciones:

1. **Crea un archivo de datos en Outlook:** el desarrollo de este software usa el SDK de Office, lo que significa que para que funcione, debe estar instalado Microsoft Office en el computador o servidor donde se ejecute este software. Lo que hace nuestro aplicativo es usar un archivo de datos de Outlook para cargar allí el PST y poder ejecutar acciones

programáticas sobre él.

2. **Extrae los items del PST:** programáticamente se empieza a recorrer el archivo PST extrayendo los items uno a uno y almacenándolos en un archivo local con formato JSON. Esta extracción va a permitir enviar los datos a la API, ya que la API solamente soporta el formato JSON.
3. **Envía los items en formato JSON a la API:** se cargan los items en modo BULK y se envían de manera cifrada a la API, la cual, recibe los datos, los procesa y por cada item genera un resultado basado en el algoritmo del triángulo del fraude.
4. **Recibe los datos de la API:** al terminar el procesamiento del lado del servidor, éste devuelve también en formato JSON los resultados, que son almacenados también en un archivo JSON de manera local ya preparados y formateados para la siguiente fase de generación del reporte.
5. **Genera un reporte en Excel:** se recorre el archivo JSON con los resultados de la API y se va llenando un archivo en Excel que previamente hemos creado como plantilla, con las alertas que el algoritmo del triángulo del fraude haya encontrado. Para este proceso, también se requiere el SDK de Microsoft Office.

Procesar un archivo PST de 1 GB puede tardar hasta 10 minutos para el algoritmo del triángulo del fraude y hasta 50 minutos si se activa la IA generativa FraudGPT.

**The Fraud Explorer** es un software que junto con **FraudGPT** detecta el fraude y la corrupción en las organizaciones

# Análisis de la voz en un IPPBX

The Fraud Explorer está en capacidad de procesar y hacer analítica del triángulo del fraude sobre conversaciones por voz a través de Asterisk. A continuación se mostrará cómo se realiza una prueba de concepto para ésta integración con la API.

Siga los pasos a continuación en Linux para realizar una instalación de Asterisk con el propósito de hacer la prueba de concepto. Si usted ya tiene un PBX operativo, sátese esta sección y continúe con el procedimiento del DIALPLAN y la instalación del endpoint y el procesador ASR.

```
# yum -y groupinstall 'Development Tools'
# yum -y install libedit libedit-devel sox mpg123
# yum install wget ssh ncurses ncurses-devel uuid uuid-devel libuuid-devel jansson-devel libxml2-devel sqlite-devel
```

Descargue Asterisk y compílelo:

```
# cd ~
# wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-17-current.tar.gz
# cd /usr/src
# tar zxvf ~/asterisk-17-current.tar.gz
# cd asterisk-17*
# ./configure --with-jansson-bundled
# make menuselect
# make
# make install
# make config
# make install-logrotate
# make samples
# systemctl enable asterisk
# systemctl restart asterisk
```

Configure el firewall para permitir conexiones TCP/UDP hacia el puerto 5060 de esta máquina virtual. Así mismo, redirija todo el tráfico RTP de los puertos UDP del 10000 al 20000 a esta máquina.

Descargue Google ASR para Asterisk y configúrelo en su PBX así:

```
# wget https://github.com/zaf/asterisk-speech-recog/tarball/master
# yum install flac flac-devel perl perl-JSON perl-libwww-perl.noarch perl-LWP-Protocol-https perl-Crypt-SSLeay
```

Copie el archivo **speech-recog.api** al directorio **/var/lib/asterisk/agi-bin** y establezca permisos de ejecución sobre el mismo. Configure ahora el DIALPLAN en el archivo **/etc/asterisk/extensions.conf** así:

```
[google-asr]
exten => 666,1,Answer();
same => n,agi(googletts.agi,"Hola, dime lo que sientes y termina con la tecla numeral",es);
same => n,agi(speech-recog.agi,es);
same => n,agi(googletts.agi, "Feliz dia, hasta luego",es);
same => n,Verbose(1,You said: ${utterance});
same => n,agi(thefraudexplorer.agi,"${utterance}");
same => n,Verbose(1,The Fraud Explorer status: ${ftaStatus});
same => n,Hangup();
```

En el archivo **/etc/asterisk/sip.conf** cree una extensión de prueba:

```
[100]
type=friend
secret=mysecret
username=100
callerid="benjamin@thefraudexplorer.com" <100>
host=dynamic
context=google-asr;
nat=force_rport,comedia
```

En el [Google Cloud API](#) habilite la característica **Google Cloud Speech API**. Copie su API Key y péguela en el archivo de configuración **/var/lib/asterisk/agi-bin/speech-recog.api** en la variable **key**. Descargue luego el [motor TTS de Google](#), descomprímalo y copie el archivo **googletts.agi** al directorio **/var/lib/asterisk/agi-bin**.

Ingresa a la plataforma web de The Fraud Explorer y entre al módulo de generación de endpoints. Allí seleccione **Asterisk VoIP PBX**, estableciendo la dirección del servidor y las credenciales API REST. Al descargar, copie el archivo resultante generado **thefraudexplorer.agi** y cópielo a la carpeta **/var/lib/asterisk/agi-bin/**.

## Así luce el código que implementa la API

Este código usa la API **endPoint?query=phrases&id=agentID** para el envío de datos semánticos. Debe ajustar al final del archivo los datos de conexión a la API, como usuario,

contraseña y otros datos propios de la instancia, como las llaves de cifrado.

```
<?php

/*
 * This function encrypts string with AES-128 bits
 *
 * @param string $unencrypted -> text to cipher
 * @param string $cipherkey -> cipher key
 *
 * return : encrypted string
 */

function encRijndael($unencrypted, $cipherkey)
{
    $key = $cipherkey;
    $iv = $cipherkey;
    $iv_utf = mb_convert_encoding($iv, 'UTF-8');
    $storeturn = mcrypt_encrypt(MCRYPT_RIJNDAEL_128, $key, $unencrypted, MCRYPT_MODE_CBC, $iv_utf);
    $storeturn = base64_encode($storeturn);

    return $storeturn;
}

/*
 * This function decrypts string with AES-128 bits
 *
 * @param string $encrypted -> text to decrypt
 * @param string $cipherkey -> cipher key
 *
 * return : decrypted string
 */

function decRijndael($encrypted, $cipherkey)
{
    $encrypted = rawurldecode($encrypted);
    $key = $cipherkey;
    $iv = $cipherkey;
    $iv_utf = mb_convert_encoding($iv, 'UTF-8');
    $storeturn = mcrypt_decrypt(MCRYPT_RIJNDAEL_128, $key, base64_decode(str_replace("_","/"),str_replace("-
```

```

", "+" , $encrypted))), MCRYPT_MODE_CBC, $iv_utf);
    $storeturn = filter_var($storeturn, FILTER_SANITIZE_STRING, FILTER_FLAG_STRIP_LOW);
    return $storeturn;
}

/*
 * This function heartbeats endpoint
 *
 * @param string $agentID -> endpoint ID
 * @param string $serverTFE -> server address
 * @param string $agentVersion -> agent version
 * @param string $keyPass -> server password
 * @param string $domain -> endpoint domain
 * @param string $cipherKey -> cipher key
 *
 * return : void function
 */

function reportOnline($agentID, $serverTFE, $agentVersion, $keyPass, $domain, $cipherKey)
{
    $rawURL = $serverTFE."/update.php";
    $pbxVersion = shell_exec("/sbin/asterisk -rx \"core show version\" | grep \"Asterisk\" | awk '{ print $2 }'");
    $pbxVersion = trim(preg_replace('/\s+/', "", $pbxVersion));
    $unwanted_chars = array('+'=>'-', '/'=>'_');
    $params = "";

    $getRequest = array(
        'token' => encRijndael($agentID, $cipherKey),
        's' => encRijndael($pbxVersion, $cipherKey),
        'v' => encRijndael($agentVersion, $cipherKey),
        'k' => encRijndael($keyPass, $cipherKey),
        'd' => encRijndael($domain, $cipherKey)
    );

    foreach($getRequest as $key=>$value) $params .= $key.'='.$value.'&';

    $params = trim($params, '&');
    $params = strtr($params, $unwanted_chars);
    $ch = curl_init();

```

```

curl_setopt($ch, CURLOPT_URL, $rawURL.'?'.$params );
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_CONNECTTIMEOUT, 60);
curl_setopt($ch, CURLOPT_USERAGENT, "Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)");
curl_setopt($ch, CURLOPT_HEADER, 0);

$result = curl_exec($ch);
curl_close($ch);
}

/*
 * This function sanitizes string for unwanted characters
 *
 * @param string $rawPhrase -> string to sanitize
 *
 * return : sanitized string
 */

function phraseSanitization($rawPhrase)
{
    $unwanted_chars = array('Š'=>'S', 'š'=>'s', 'Ž'=>'Z', 'ž'=>'z', 'À'=>'A', 'Á'=>'A', 'Â'=>'A', 'Ã'=>'A', 'Ä'=>'A',
'Å'=>'A', 'Æ'=>'A', 'Ç'=>'C',
    'È'=>'E', 'É'=>'E', 'Ê'=>'E', 'Ë'=>'E', 'Ì'=>'I', 'Í'=>'I', 'Î'=>'I', 'Ï'=>'I', 'Ñ'=>'N', 'Ò'=>'O', 'Ó'=>'O', 'Ô'=>'O',
'Õ'=>'O', 'Ö'=>'O',
    'Ø'=>'O', 'Ù'=>'U', 'Ú'=>'U', 'Û'=>'U', 'Ü'=>'U', 'Ý'=>'Y', 'Þ'=>'B', 'ß'=>'Ss', 'à'=>'a', 'á'=>'a', 'â'=>'a',
'ã'=>'a', 'ä'=>'a', 'å'=>'a',
    'æ'=>'a', 'ç'=>'c', 'è'=>'e', 'é'=>'e', 'ê'=>'e', 'ë'=>'e', 'ì'=>'i', 'í'=>'i', 'î'=>'i', 'ï'=>'i', 'ð'=>'o', 'ñ'=>'n',
'd'=>'o', 'ó'=>'o',
    'ô'=>'o', 'õ'=>'o', 'ö'=>'o', 'ø'=>'o', 'ù'=>'u', 'ú'=>'u', 'û'=>'u', 'ý'=>'y', 'þ'=>'b', 'ÿ'=>'y');

    $sanitizedPhrase = strtr($rawPhrase, $unwanted_chars);
    $sanitizedPhrase = strtolower($sanitizedPhrase);

    return $sanitizedPhrase;
}

/*
 * This function sends data (words) to the server
 *
 * @param string $serverTFE -> server address

```

```

* @param string $agentId -> endpoint ID
* @param string $restUser -> REST username
* @param string $restPass -> REST password
* @param string $ipAddress -> endpoint IP address
* @param string $domain -> endpoint company domain
* @param string $callWith -> call with extension
* @param string $phrases -> paragraph
*
* return : void function
*/

function sendData($serverTFE, $agentId, $restUser, $restPass, $ipAddress, $domain, $callWith, $phrases)
{
    $serverAddress = $serverTFE."/rest/endPoints?query=phrases&id=".$agentId;
    $APIuser = $restUser;
    $APIpass = $restPass;

    $postRequest = array(
        'hostPrivateIP' => $ipAddress,
        'userDomain' => $domain,
        'appTitle' => $callWith,
        'phrases' => $phrases
    );

    $payload = json_encode($postRequest);

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $serverAddress);
    curl_setopt($ch, CURLOPT_POST, 1);
    curl_setopt($ch, CURLOPT_POSTFIELDS, $payload);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);

    $headers = [
        'username: ' . $APIuser,
        'password: ' . $APIpass,
        'Content-Type: application/json',
    ];

    curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
    $server_output = curl_exec($ch);

```

```

    curl_close ($ch);
}

/*
 * This function reads server remote commands
 *
 * @param string $serverTFE -> server address
 * @param string $cipherKey -> cipher key
 *
 * return : command status
 */

function collectionPhraseStatus($serverTFE, $cipherKey)
{
    $xml = simplexml_load_file($serverTFE.'/update.xml');
    $phraseCollectionStatus = decRijndael($xml->token[0]['arg'], $cipherKey);

    if ($phraseCollectionStatus == "textAnalytics 1") $phraseStatus = "enabled";
    else $phraseStatus = "disabled";

    return $phraseStatus;
}

/* PBX internal variables */

$agivars = array();

while (!feof(STDIN))
{
    $agivar = trim(fgets(STDIN));

    if ($agivar === "") break;

    $agivar = explode(':', $agivar);
    $agivars[$agivar[0]] = trim($agivar[1]);
}

extract($agivars);

/* PBX Endpoint variables */

```

```

$serverTFE = "https://demo.thefraudexplorer.com";
$keyPass = "31173";
$cipherKey = "yourkeyandiv";
$restUser = "apirestuser";
$restPass = "apirestpassword";
$agentVersion = "0.1";
$ipAddress = shell_exec("/sbin/asterisk -rx \"sip show peers\" | grep \"\".$agi_callerid.\"/\".$agi_callerid.\"\" | awk '{
print $2 }'");
$ipAddress = trim(preg_replace('/\s+/', '', $ipAddress));
$callerName = $agi_calleridname;
$callerNameArray = explode("@", $callerName);
$domain = $callerNameArray[1];
$name = $callerNameArray[0];
$agentId = $name."_".$agi_callerid."1kc9_pbx";
$callWith = "Phone call with ".$agi_dnid;
$phrases = phraseSanitization($argv[1]);

/* Exit if no phrase */

if ($phrases == "" || collectionPhraseStatus($serverTFE, $cipherKey) == "disabled")
{
    echo "SET VARIABLE ftaStatus no-sent";
    exit;
}

/* Create or update endpoint and send data */

reportOnline($agentId, $serverTFE, $agentVersion, $keyPass, $domain, $cipherKey);
sendData($serverTFE, $agentId, $restUser, $restPass, $ipAddress, $domain, $callWith, $phrases);

/* Finish AGI */

echo "SET VARIABLE ftaStatus sent-ok";

?>

```

## Instale un Softphone

Descargue Bria para iOS o Android y registre la extensión 100. Cuando registre, llame al número "666" y escuchará una voz que le pide decirle "como se siente". Hable y exprese lo que siente, al finalizar presione la tecla # y luego visualice la consola **The Fraud Explorer** en busca de eventos alertas sobre las frases dichas.

## Configuración en producción

Para su sistema en producción, debe instalar el motor de Google ASR y TTS junto con el AGI de **The Fraud Explorer** y usar el módulo **ChanSpy**, para capturar el audio de una llamada en vivo y posteriormente pasar su transcripción a The Fraud Explorer. El único requisito que debe respetar es el contar con una buena identificación de las personas que están conectadas a la planta, a través del Caller ID y el User Name, puesto que de esta manera se identificarán los usuarios en la plataforma de analítica.