SPLITTER: An Approach to Difficult Correlation, Traffic-Analysis and Statistical Attacks Inside TOR Network

Rêner Alberto F. Silva (aka Gr1nch) – DcLabs Security Team
rener.silva@protonmail.com

ABSTRACT

The number of TOR NODES also know as TOR RELAYS is growing, but we can expect that the number of compromised TOR NODES/RELAYS is also growing.[16]

The list of adversaries include but not limited to: researchers, security information related companies, governments and many others. The adversaries are running compromised TOR relays to capture data in transit inside TOR network. They are collecting, analyzing and correlating the captured data aiming to identify who is using the TOR Network and which kind of sensitive data is being transferred by these users inside this anonymous network. [2, 3, 9, 10]

The results presented by Sambuddho Chakravarty, Marco V. Barbera, Georgios Portokalidis, Michalis Polychronakis and Angelos D. Keromytis in the paper:
"On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records" [1] – shows the overall accuracy of about 81.4% of success to identify the source of the TOR requests in the real-world experiments. This result is one proof about the power of traffic analysis, correlation and statistically related attacks against anonymous networks demonstrating the high probability of success to identify the source and the final destination of requests crossing the TOR network.

We have seen many cases where people using anonymous networks like TOR has been identified.[1, 11, 12, 13, 13.1, 14, 15]

The philosophical discussion behind the reason why the people decide to use anonymous networks it's beyond of the scope of this paper.[11, 12, 13, 14, 15, 19]

My research has been based on TOR network but nothing prevents that the idea behind the techniques presented here could be implemented in the context of others anonymous networks.

This paper will discuss the feasibility of exploiting the common weakness of the majority of de-anonymization techniques which is the possibility of difficult or break the correlation.

The ability of the adversary to collect, read, understand and correlate the intercepted data is essential for some de-anonymization techniques. In some cases, only identify both sides of the communication could not be enough. The adversary also need have access to the "message" and be able to understand the context of the intercepted message before performing the correlation of acquired information.

Correlation attacks, traffic-analysis attacks, and statistical attacks become more difficult in the following circumstances:

- The adversary was not able to collect data enough:
  By reducing the total amount of data that adversaries collects and the total amount of data available for analyses the adversary could not understand or be not able to associate the collected data with previous data collected.[1, 2, 3, 9, 10]

- The adversary has only one small piece of the "message":
  By dividing the total amount of data to be transmitted and spreading it through different communication channels which the adversary do not controls reduces the number of pieces collected by the adversary.[1, 2, 3, 9, 10]

- The interval between the intercepted piece of the message is too long:
  Time is a sensitive item for some de-anonymization techniques based on the correlation. If the interval between the intercepted message is too long and do not allow the adversary to associate the time between the collected pieces, the correlation could fail.[1, 2, 9]
  Also, if the collected pieces belong to different "messages", it will demand much more effort to associate and correlate the content of both parts.[9] This scenario can affect the ability of adversary in understanding the context of the message.

- The lack or loss of pattern which identifies the piece of the intercepted message:

  Usually, the adversary should assume a strategic position to collect data. Some de-anonymization techniques assume that adversary controls at least two interception points.[1, 2, 3, 6, 10]
  Some de-anonymization techniques inject or rely on patterns to identify packets crossing the TOR circuit.[1, 3, 6, 10]
  The same pattern should be observed in both point of data collection, more specific near[1, 10] or inside[6, 10] the compromised TOR Entry Node and near[1, 10] or inside[6, 10] the compromised TOR Exit Node.

- The adversary is not able to keep the compromised communication channel open:

  Keep the control over the communication channel is essential for some de-anonymization techniques.[1, 2, 4, 6, 9, 10].
  Some of these techniques relies on the fact that victim will transfer a specific amount of data trough the same communication channel during a specific amount of time.[1, 2, 4, 10]
  By changing the communication channel from one controlled by the adversary to another which one the adversary does not controls, can difficult many of these techniques. The time is a sensitive item in the equation of these techniques and the premature end of the communication channel can affect the correlation, statistics and traffic analyses.

This paper will face the challenge of difficult or in some cases break the correlation of the collected data inside[2, 4, 6, 10] or even outside[1, 4] TOR network.
As a bonus, this paper will provide a suggestion for a TOR based network infrastructure designed to provide a better performance and stability.
According to Nicholas Hopper, Eugene Vasserman and Eric Chan-Tin in the paper "How Much Anonymity does Network Latency Leak?"[4] the stability and performance of TOR circuits can be directly affected depending on how much the current users of these TOR circuits are overloading the TOR Network.

This paper will propose a better load balancing for the TOR network. The initial results of my stability and speed tests show the possibility of watching High Definition (1080p 60fps) videos from Youtube or related media stream services, without interruptions even changing the TOR circuits every 10s, splitting and sending data through different countries, destroying and recreating the TOR instances every 2 minutes.
The results of my tests show that after applying the SPLITTER approach, the overall average that adversaries would be able to collect is only 0.5% for each compromised TOR EXIT NODE.

INTRODUCTION

TOR is an anonymizing peer-to-peer network which supports the anonymous transport of TCP streams over the internet using low-latency TCP tunnels, also called TOR CIRCUITS.[7, 8]

The TOR Hidden Service allows internet services like websites, blogs, chat, file sharing among others whose authors are at risk of legal prosecution still operating using pseudonymous. [11, 12, 17]

The official TOR description is:

> "Tor is free software and an open network that helps you defend against traffic analysis, a form of network surveillance that threatens personal freedom and privacy, confidential business activities and relationships, and state security." [7]
>
> "The Tor network is a group of volunteer-operated servers that allows people to improve their privacy and security on the Internet. Tor's users employ this network by connecting through a series of virtual tunnels rather than making a direct connection, thus allowing both organizations and individuals to share information over public networks without compromising their privacy. Along the same line, Tor is an effective censorship circumvention tool, allowing its users to reach otherwise blocked destinations or content. Tor can also be used as a building block for software developers to create new communication tools with built-in privacy features." [8]
>
> "Tor was originally designed, implemented, and deployed as a third-generation onion routing project of the Naval Research Laboratory. It was originally developed with the U.S. Navy in mind, for the primary purpose of protecting government communications. Today, it is used every day for a wide variety of purposes by the military, journalists, law enforcement officers, activists, and many others. ". [19]

Despite the poor network performance of many TOR relays, this network remains very suitable for common tasks like web browsing.

The communication inside TOR networks takes place using TOR circuits.[8]

A single TOR circuit is composed by:

- TOR Entry Node
- TOR Middle Node
- TOR Exit Node

A short and not exhaustive description about the TOR circuit build process is:

1. The TOR client obtains a list of TOR relays from a directory server.

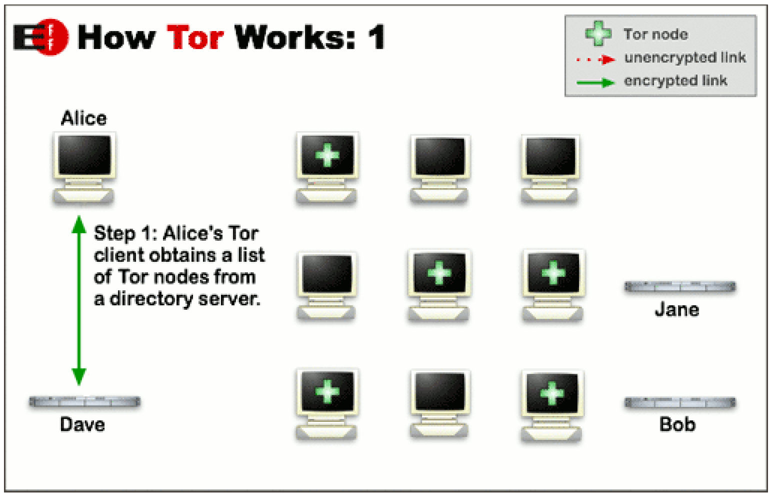

How Tor Works: 1
Tor node
unencrypted link
encrypted link
Alice
Step 1: Alice's Tor client obtains a list of Tor nodes from a directory server.
Jane
Dave
Bob

Image 1 - Source: Official TOR Web Site [8]

2. The TOR client picks a random path to destination server:

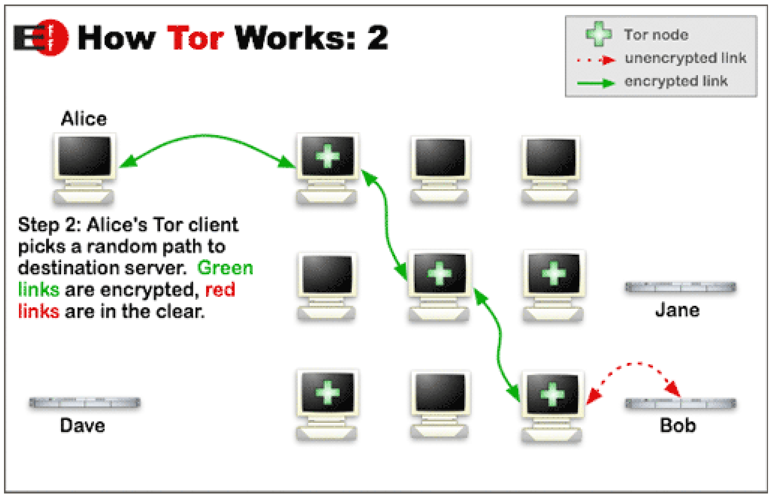

How Tor Works: 2
Tor node
unencrypted link
encrypted link
Alice
Step 2: Alice's Tor client picks a random path to destination server. Green links are encrypted, red links are in the clear.
Jane
Dave
Bob

Image 2 - Source: Official TOR Web Site [8]

3. By default TOR re-utilizes the same circuit during 10 minutes. After this time a new circuit is created.

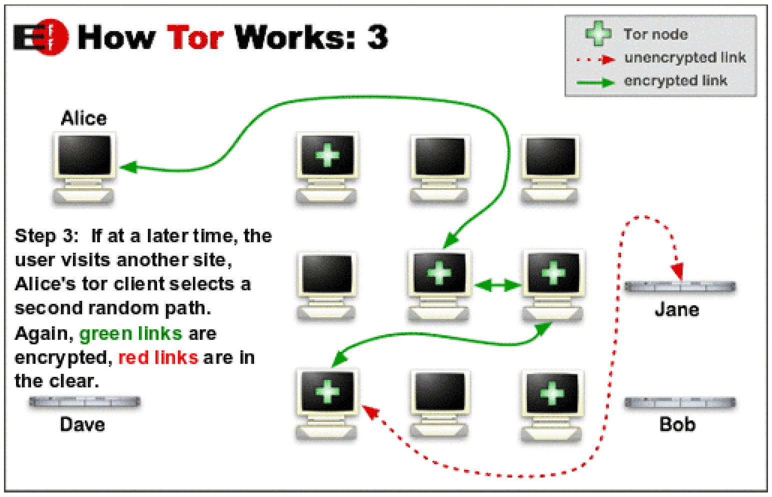

How Tor Works: 3
Tor node
unencrypted link
encrypted link
Alice
Step 3: If at a later time, the user visits another site, Alice's tor client selects a second random path. Again, green links are encrypted, red links are in the clear.
Jane
Dave
Bob

Image 3 - Source: Official TOR Web Site [8]

Initial Considerations about the TOR CIRCUIT:

During the second step of the TOR circuit construction process, TOR selects a random path for the TOR circuit. In other words, TOR will select a random combination of the ENTRY NODE, MIDDLE NODE, and EXIT NODE.

ENTRY NODE:

This relay only knows the TOR user machine and the TOR MIDDLE NODE.

As the ENTRY NODE is the first hop inside TOR network, it assumes a very important role due to the fact the TOR user or victim connects to TOR ENTRY NODE and if the adversary controls the TOR ENTRY NODE it means that adversary already knows the IP address from the victim.

To take the control of the ENTRY NODE or take the control of routers near the entry node is essential for the adversary.

The strategical position of ENTRY NODE makes it a very valuable target and an important item for many de-anonymization techniques.[1, 2, 3, 4, 5, 6, 10, 23]

MIDDLE NODE:

This relay only knows the ENTRY NODE and the EXIT NODE. This relay doesn't know the TOR user IP or the origin from packets, also the MIDDLE NODE doesn't know the final destination of packets. However, this node is very important and valuable item for the TOR users because it's responsible to break the natural correlation between the TOR user and the final destination of packets. Providing this way the anonymity behind the TOR network. [3, 7, 8, 20]

Attacks targeting this TOR NODE are not common[23] but Rob Jansen, Marc Juarez, Rafa Gálvez, Tariq Elahi and Claudia Diaz shows in the paper "Inside Job: Applying Traffic Analysis to Measure Tor from Within" [20] , the possibility to use a fingerprinting methodology to discover the popularity of onion hidden services[17, 18], filtering and targeting specific nodes in TOR network suck particular TOR guard[21, 22] relays.

EXIT NODE:

This relay only knows the MIDDLE NODE and the final destination of packets. This relay don't know the TOR user IP address and also don't know the ENTRY NODE but if the adversary controls the EXIT NODE he knows the final destination of packets.

The strategical position of EXIT NODE makes it a very valuable target and an important item for de-anonymization techniques.[1, 2, 3, 4, 5, 6, 10, 23]

Threat modeling of TOR de-anonymization techniques:

In summary, the essence behind of de-anonymization techniques is the feasibility of identifying the origin and destination of packets in transit inside or crossing the anonymous networks. [1, 2, 3, 4, 5, 6, 10, 18]

Increasing the possibility of allowing a direct identification of the person who is using the anonymous network.

As a bonus, de-anonymization techniques could provide to the adversary the possibility of understanding the content and context of intercepted data, allowing the attacker to take strategical decisions for further investigation or attacks. [11, 12, 13, 13.1, 14, 15]

Anonymous networks are based on common internet protocols like TCP/IP and TOR network also uses TCP/IP as network transport protocol, using TLS encryption to prevent the attacker to intercept the data in transit between the TOR NODES.[23, 25, 42]

The majority of attacks against the anonymous networks take place on the NODES or RELAYS which compose this network. However only intercept the packets is not enough, complex and more sophisticated techniques came along the years and are being applied together to break the anonymity.[23]

As a result of the effort applied the attacker can take decisions based on the accuracy of results provided by the applied de-anonymization techniques. [11, 12, 13, 13.1, 14, 15, 19]

The image below shows the traditional representation of a compromised TOR circuit where the adversary is able to observe the traffic crossing the ENTRY NODE and the EXIT NODE.

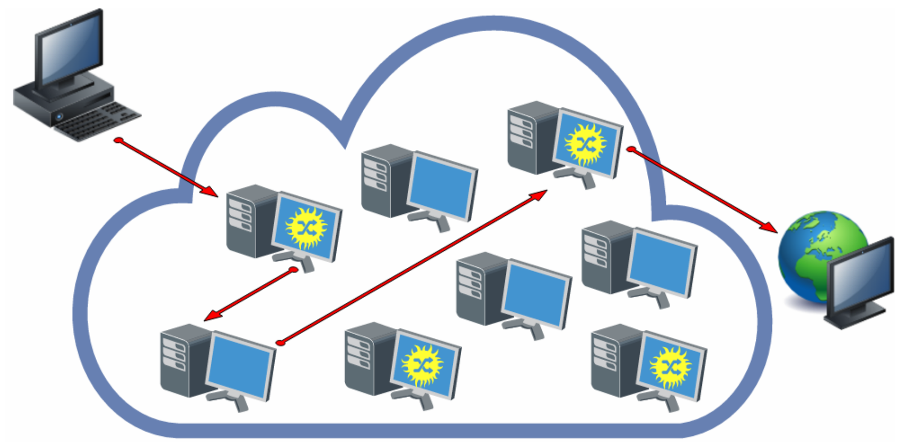


Image 4 – Compromised TOR circuit.

Due to the lack of a natural and direct correlation between TOR ENTRY NODE and TOR EXIT NODE the majority of de-anonymization techniques are based on the feasibility of performing a very organized and strategical combination of traffic analysis attack, statistical attacks and correlation attacks.

These powerful attacks work together using the intercepted data inside or near of compromised EXIT NODE; with the intercepted data inside or near of compromised ENTRY NODE to allow further association, correlation and statistical analyses.

The image below shows the traditional representation of a compromised TOR circuit where the adversary is able to observe the traffic crossing the ENTRY NODE and the EXIT NODE.

<u>The complex process behind de-anonymization techniques:</u>

A generalist and not exhaustive description about the complete de-anonymization process which allows the attacker take decisions based on results of the post-traffic-analysis, data correlation and statistical analyses applied against the intercepted and de-anonymized traffic data, could be based on two important pillars:

1. The ability of attacker in identify with precision the source and destination of a reasonable number packets or request sent or received by the target victim. Time correlation could be applied in some de-anonymization techniques at this point.

2. A specific amount of data could be necessary to confirm the victim identity and each technique applied will request a different amount of data to match the pattern. [1, 2, 3, 4, 5, 6, 10, 23].
   However only identify the origin and destination of a few numbers of packets could not be enough. Depending on the context and the propose of the attack, the adversary still needs to repeat the attack to collect more data and keep the monitoring of the compromised communication channel to collect more data before taking decisions. The decisions will lead the attacker to his final goal or to his next attack. [11, 12, 13, 13.1, 14, 15]

To accomplish this complex task, some de-anonymization techniques will promote a network disturb, injecting or creating a unique and specific pattern in the network flow or packets.
Some situations could create a natural pattern. This pattern is created by the packets flow crossing the compromised TOR circuit. By analyzing the behavior, pieces of information or characteristics of TOR network related equipment the attacker can identify a pattern and use this specific pattern to identify the victim.[1, 2, 4, 5, 6, 10, 18].

The adversary can analyses, for example, the physical aspects of the anonymous network related equipment and it's behavior. The temperature of the CPU has been observed and used in the past to identify hidden services inside the TOR network. [17]

According to Steven J. Murdoch "while handling with the total throughput of the network would still affect the load of CPU and thus heat output".

In this specific technique the temperature of CPU can be observed and impacts in the timestamps due to the influence of device temperature which affects the crystal oscillator driving the system clock. According to Steven J. Murdoch, the attacker can measure this effect remotely by requesting timestamps and deriving clock skew.[18]

In the scenario of the paper, the elevation of the CPU's temperature reflects on timestamps, creating a pattern and could result in the identification of hidden services.[17]

For the majority of de-anonymization techniques, the same pattern should be observed in two different data collection points controlled by the adversary:

- Inside the compromised TOR EXIT NODE [2, 4, 5, 6, 10], or in a network related equipment near of the compromised TOR EXIT NODE[1, 4, 18]. Some techniques can use compromised web servers to inject a pattern or just correlate the injected pattern.

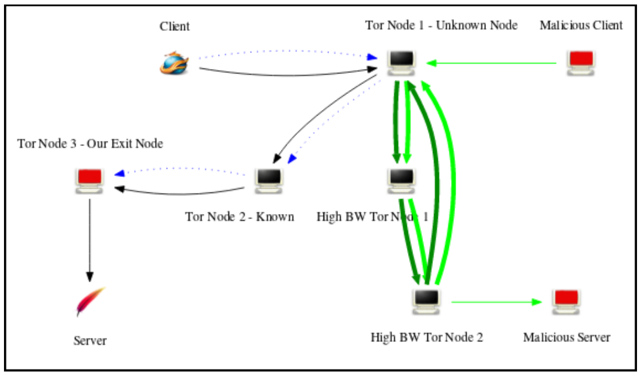


Image 5: Source – Paper "A Practical Congestion Attack on Tor Using Long Paths"[2]

- Inside the compromised TOR ENTRY NODE [2, 4, 5, 6, 10], or in a network related equipment near of the compromised TOR ENTY NODE[1, 4, 18]. Sambuddho Chakravarty, Marco V. Barbera, Georgios Portokalidis, Michalis Polychronakis and Angelos D. Keromytis have demonstrated in the paper "On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records"[1], how the adversary can use a compromised hidden service[17] or even a normal web server to deliberately perturb the characteristics of user traffic at the server side, and observing a similar perturbation of traffic in the client side through statistical correlation. In this technique, they are looking for the pattern outside of TOR network and following this pattern until the victim. According to the authors of this paper, this technique presented 81.4% of overall de-anonymization accuracy in the real-world experiments.

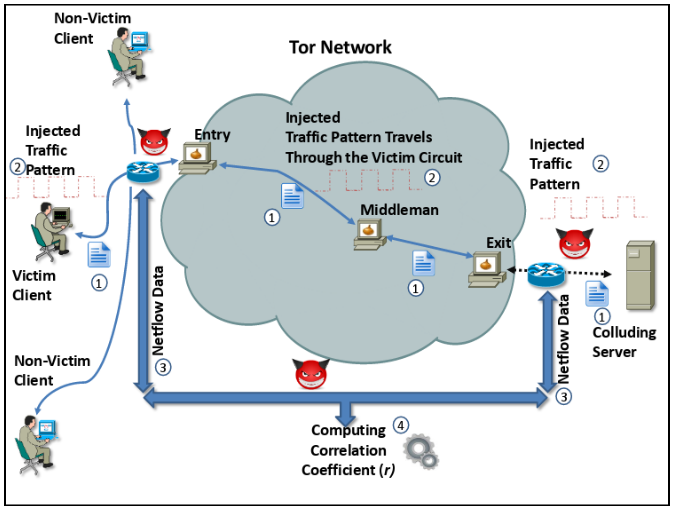


Image 6 source: "*On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records*"[1]

Independent of which technique the adversary choose for attack, once observed the occurrence of the specific pattern, inside[2, 4, 5, 6, 10] or near[1, 4, 18] of TOR EXIT NODE, the adversary is able to associated it with the same pattern previously observed inside[2, 4, 5, 6, 10] or near[1, 4, 18] of TOR ENTRY NODE.

A successful correlation and association between TOR ENTRY NODE and TOR EXIT NODE breaks the natural anonymity provide by TOR network, giving the attacker the opportunity to take analyze the victim's captured data decisions about the next phase of attack.

COMMON WEAKNESS OF DE-ANONYMIZATION TECHNIQUES

Even applying advanced and complex statistical analyses, the majority of de-anonymization techniques assumes that the victim will transfer the total amount of data necessary to identify the pattern using:

- The same TOR circuit.
- The same global network path from the compromised web server or compromised EXIT NODE to the client.
- Also, techniques which require a cert amount of data to transmit the pattern assumes that the same TCP connection stream will still active during a specific period of time.

Some de-anonymization techniques can fail considering the feasibility of anonymous network users disturb the natural behavior of TOR network, web browsers behavior and established TCP stream. [1, 2, 4, 5, 6, 10, 18]
Users actions aiming to change the natural behavior could directly affect the adversary ability of:

- Collect data.
- Correlate the collected data with the time.
- Identify the pattern.

Many de-anonymization techniques rely on the default TOR CIRCUIT life period, which by default is 10 minutes. [8, 26, 27]
It means that after establishing a TOR circuit, this connection will remain open for 10 minutes and HTTP request or TCP packets send or received by the victim during this period of time is prone to be intercepted and analyzed by compromised TOR NODES or network routers under control of the adversary. [1, 2, 4, 5, 6, 10, 18]

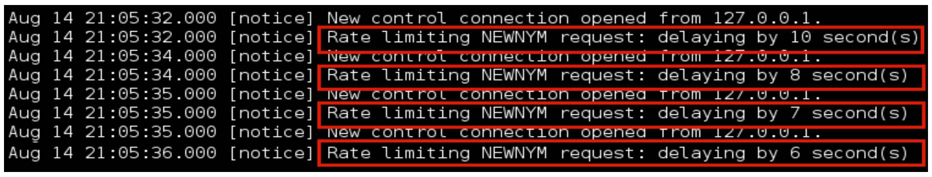
Another important fact is, TOR does not transfer a TCP stream from a circuit to another. It means that persistent TCP stream connections will remain established using the same circuit.[42] De-anonymization techniques take advantage of this tricking the victims to perform a huge download using the same TCP stream and consecutively the same TOR circuit for a specific amount of time necessary to transfer the injected pattern.[1]

If the user, disturb the connection forcing the premature death of TCP stream, and TOR circuit, the adversary ability of collect data, correlate and identify the pattern is compromised and the technique can generate a false positive due to the low amount of data collected. [1, 2, 4, 5, 6, 10, 18]

Another feasible vulnerability arises when the adversary has a limited number of compromised TOR NODES or any other limited number of data collection points.[1, 2, 4, 5, 6, 10, 18]
If the user disturbs the natural TOR circuit lifetime, reducing the lifetime from **10 minutes** to the lowest accepted value which is **10 seconds**, the new TOR circuit could select a random circuit path which does not use the adversary's compromised nodes. The next requests or packets sent through this TOR INSTANCE will be send using a channel which the adversary does not controls, reducing his ability to collect and correlate the intercepted data.
However, this TOR instance running and changing the circuits every 10 seconds, can generate in the future a new TOR circuit which is under the adversary's control, giving him the opportunity to observe the network traffic again. Beyond the possibility of reducing the total amount of data collected by the adversary, by disturbing the TOR circuit lifetime, the user can affect the adversary ability to detect the required pattern.

The effectiveness of this evasion technique is questionable, because an adversary which controls a large number of compromised TOR NODES or compromised routers have more chances of still observing the traffic even after the user force the new circuit. However, more effort for correlation will be required and we discuss this scenario later.

The image below shows one attempt to force a new TOR circuit after 2 seconds of a new circuit being created:

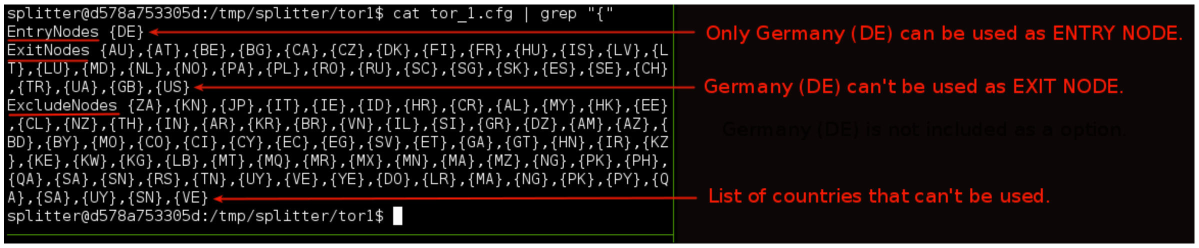


```
Aug 14 21:05:32.000 [notice] New control connection opened from 127.0.0.1.
Aug 14 21:05:32.000 [notice] Rate limiting NEWNYM request: delaying by 10 second(s)
Aug 14 21:05:34.000 [notice] New control connection opened from 127.0.0.1.
Aug 14 21:05:34.000 [notice] Rate limiting NEWNYM request: delaying by 8 second(s)
Aug 14 21:05:35.000 [notice] New control connection opened from 127.0.0.1.
Aug 14 21:05:35.000 [notice] Rate limiting NEWNYM request: delaying by 7 second(s)
Aug 14 21:05:35.000 [notice] New control connection opened from 127.0.0.1.
Aug 14 21:05:36.000 [notice] Rate limiting NEWNYM request: delaying by 6 second(s)
```

Image 7 – An consecutively attempt to create a new TOR circuit with the interval under 10 seconds.

If the user is able to disturb the natural selection of global network path for TCP packets traveling from his machine to the destination machine, by forcing a new TOR CIRCUIT or forcing the packets to be routed through an unexpected global network path. Depending on the de-anonymization technique applied, the adversary can lose his ability to collect data and the pattern can be lost during the transportation due to the difference from different transport protocols network and types of equipment around the world.[1, 25, 28, 29, 32]

The feasibility of this evasion arises when the TOR user has the opportunity to control the geolocation of TOR ENTRY NODE, and TOR EXIT NODE. [8, 27]



```
splitter@d578a753305d:/tmp/splitter/tor1$ cat tor_1.cfg | grep "{"
EntryNodes {DE}
ExitNodes {AU},{AT},{BE},{BG},{CA},{CZ},{DK},{FI},{FR},{HU},{IS},{LV},{L
T},{LU},{MD},{NL},{NO},{PA},{PL},{RO},{RU},{SC},{SG},{SK},{ES},{SE},{CH}
,{TR},{UA},{GB},{US}
ExcludeNodes {ZA},{KN},{JP},{IT},{IE},{ID},{HR},{CR},{AL},{MY},{HK},{EE}
,{CL},{NZ},{TH},{IN},{AR},{KR},{BR},{VN},{IL},{SI},{GR},{DZ},{AM},{AZ},{
BD},{BY},{MO},{CO},{CI},{CY},{EC},{EG},{SV},{ET},{GA},{GT},{HN},{IR},{KZ
},{KE},{KW},{KG},{LB},{MT},{MQ},{MR},{MX},{MN},{MA},{MZ},{NG},{PK},{PH},
{QA},{SA},{SN},{RS},{TN},{UY},{VE},{YE},{DO},{LR},{MA},{NG},{PK},{PY},{Q
A},{SA},{UY},{SN},{VE}
splitter@d578a753305d:/tmp/splitter/tor1$
```

Only Germany (DE) can be used as ENTRY NODE.
Germany (DE) can't be used as EXIT NODE.
Germany (DE) is not included as a option.
List of countries that can't be used.

Image 8 – A screenshot showing how the SPLITTER uses GeoIP to manipulate the TOR circuit in the TOR config file.[27]

The user can control the continent from where his connection with TOR ENTRY NODE will be originated by using VPS, VPN or the combination of both.[30, 31]

Considering this scenario, even after discovering the source of requests, the adversary, should apply more effort to discover who was connected on this VPS or VPN server on that specific time because the discovered IP could belong to a public VPN service.

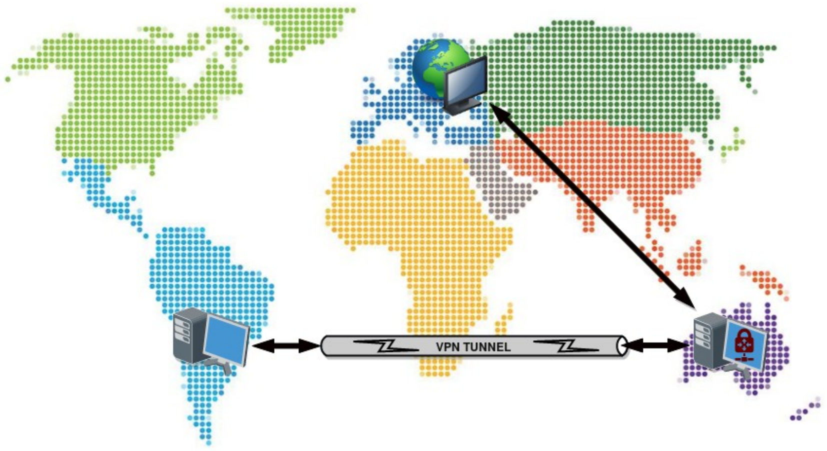


Image 9: A conventional representation of VPN connection

The Exploit Challenge:

In order to abuse and take advantage of all common de-anonymization techniques' weakness, the anonymous network user should have the ability to enforce for every single TCP stream, requests or packets transmitted:

1. A different TOR circuit from the previous requests. It means, ensures that the ENTRY NODE or the EXIT NODE will be different from the previous request.

2. A different global network path for packets traveling from his machine, crossing the anonymous network and arriving in the final destination server.

3. Control or disturb the time which TOR could generate the same compromised TOR CIRCUIT again.

4. Disturb the TCP stream lifetime, interrupting the transmission if the stream is being used for more than "X" minutes.

THE SPLITTER SOLUTION

To exploit the common weakness of TOR related de-anonymization techniques, difficulty traffic-analysis, correlation and statistically related attacks on the TOR network. [1, 2, 3, 4, 5, 6, 10, 20, 23, 28]
I developed a free open-source TOR network based script called SPLITTER. This script configures and applies a systematic chain of free open-source solutions, working together to difficult the TOR related de-anonymization techniques and ensure a better performance for TOR network. The result is a better TOR user experience and a more secure TOR network related connection approach.
The SPLITTER is licensed under the BSD - License and was created with an initial academic propose.[41]
The user accepts the total responsibility for his acts while using this tool.

For the best effectiveness of the theoretical approach behind the SPLITTER solution, a low-cost private VPS and VPN networks chain should be considered. The idea behind this globally distributed network infrastructure is difficult more specific traffic-analysis attacks and do not allow a direct association between the TOR network and the user. This network approach will be called "SPLITTER NETWORK" in this paper and comprehends few VPS machines under the control of the user running the SPLITTER script but using a public VPN service to connect in TOR network.

The bundle of linux open-source tools which compose the SPLITTER tool are:

1. **HAPROXY Community Edition:** "HAProxy is a free, very fast and reliable solution offering high availability, load balancing, and proxying for TCP and HTTP-based applications. It is particularly suited for very high traffic web sites and powers quite a number of the world's most visited ones. Over the years it has become the de-facto standard opensource load balancer, is now shipped with most mainstream Linux distributions, and is often deployed by default in cloud platforms. Since it does not advertise itself, we only know it's used when the admins report it." [33]

2. **PRIVOXY:** "Privoxy is a non-caching web proxy with advanced filtering capabilities for enhancing privacy, modifying web page data and HTTP headers, controlling access, and removing ads and other obnoxious Internet junk. Privoxy has a flexible configuration and can be customized to suit individual needs and tastes. It has application for both stand-alone systems and multi-user networks."[34]

3. **TOR (standalone):** The TOR network client.[35]

SPLITTER overview:

Each SPLITTER related tool is applied in a systematic sequence, driving the TCP packets from the user browser or application, first to HAPROXY, second to PRIVOXY and the last step is the TOR standalone which provide the connection with TOR network. After being routed through the current active TOR circuit[8], the packet reaches the final destination. The answer for this TCP packet will follow the reverse path.

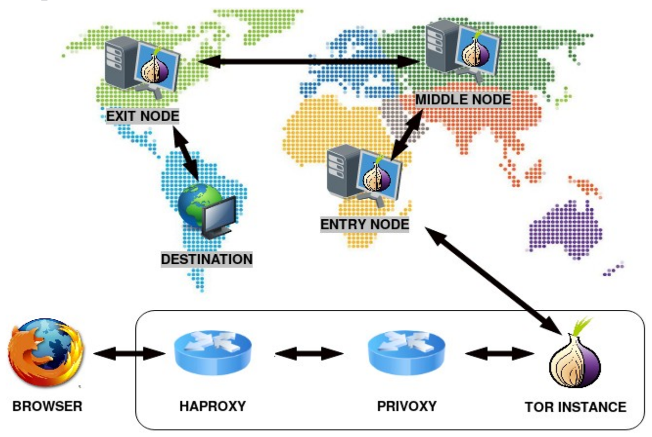


Image 10: The TCP packet path in the SPLITTER context.

The SPLITTER will create and handle with many TOR network connections. A single TOR standalone network connection is also called in this paper as “TOR INSTANCE” and comprehends a single and unique execution of TOR standalone running and administrating it’s own TOR network circuits.[8, 16, 21, 22, 24, 25, 26, 27]

The SPLITTER gives the user the opportunity to configure every single parameter related to the execution of HAPROXY, PRIVOXY, and TOR standalone. [27, 36, 37]

However, the most important aspect of this tool is the geolocation approach and how it selects the countries which will be enforced to compose the TOR circuit.[8, 16, 21, 22, 24, 26, 27, 29]

The user should define how many TOR instances per country and how many countries the SPLITTER can use. It’s possible for example to create a number “***X***” of instances using the same country, as ENTRY NODE or EXIT NODE.

TOR instances load balance overview:

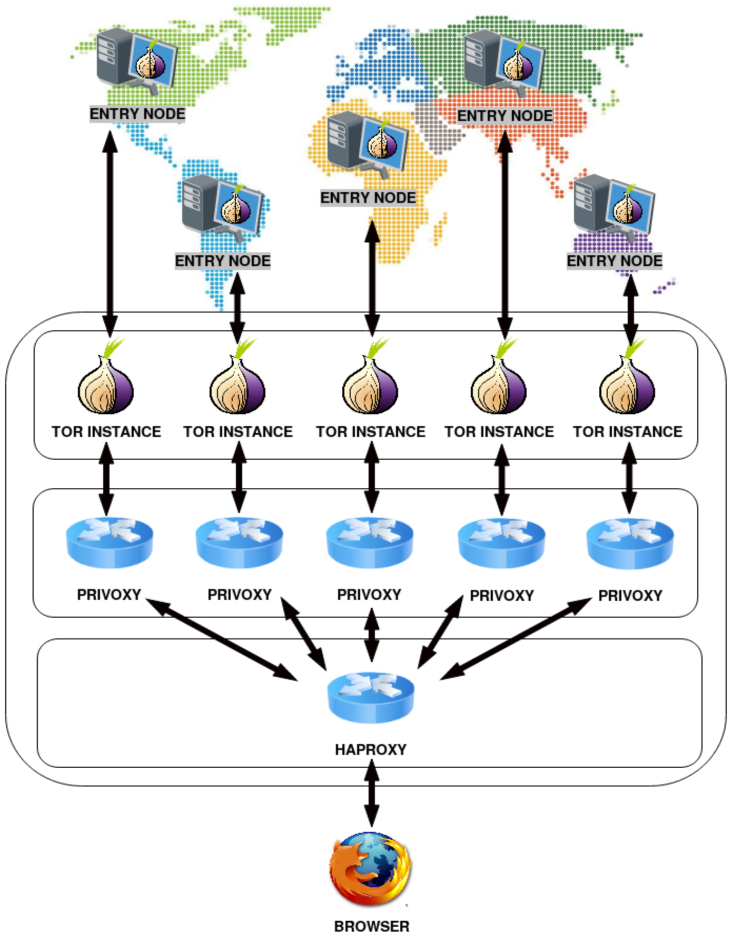


Image 11: A global representation of SPLITTER load balance considering the ENTRY NODE enforcement.

Considering a single TOR instance, by default the SPLITTER will never use the same country as TOR ENTRY NODE and TOR EXIT NODE. This rule forces the same adversary compromise TOR nodes in different countries to be able to capture and correlate the user data transmitted using the currently active and selected TOR circuit.

Default "Anti-Correlation" rules:

1. Always select a random country, from the list of countries that user accepts use as TOR ENTRY node or TOR EXIT node depending on which TOR node the user decide to enforce. It means that all random TOR circuits created by this manipulated TOR instance have a great chance to have a unique geolocation oriented combination of TOR ENTRY NODE and TOR EXIT NODE. This feature can by default difficult the correlation of many de-anonymization techniques based on:
   A) The absence of adversary's compromised TOR nodes or compromised network related equipment in both randomly selected countries.[1, 2, 3, 4, 5, 6, 10, 20, 23, 28]
   B) The deliberated disturbed created by SPLITTER in the natural global network path for packets in transit between the user machine and the destination server. [1, 2, 3, 4, 5, 6, 10, 20, 23, 28]

2. Considering the natural random country selection of TOR algorithm[8] which inside the SPLITTER manipulated context, will compose the beginning or the end of the TOR circuit, depending on which node/relay the user decide to enforce.[8] The probability exists for future TOR circuits[8] created by this TOR instance, select once again the same previous combination of TOR ENTRY node and TOR EXIT used by this TOR instance in the past. Aiming to reduce this risk, the SPLITTER also controls the life circle of the TOR instance, giving the user the control about how long time a TOR INSTANCE can remain alive enforced to use a specific country as ENTRY NODE or EXIT NODE. As result:
   A) This rule affects the random geolocation[29] oriented combination of TOR ENTRY NODE and TOR EXIT NODE.
   B) This rule disturbs the lifetime of TCP streams interrupting the TCP streams associated with this TOR instance when longer than "***X***" minutes. The premature interruption of an established TCP stream can affect the ability of the adversary to transmitting the pattern depending on the de-anonymization technique. [1, 2, 3, 4, 5, 6, 10, 20, 23, 28]

The life circle of a single TOR INSTANCE inside the SPLITTER context comprehends:

1. After selecting a random new country, the SPLITTER will write the TOR configuration file based on the TOR options[27] defined by the user. By default, the first SPLITTER's rule will be always respected. However, there are two exceptions to the first default rule:
   A) When the user decides to work with SPLITTER SPEED MODE described later in this paper. In this context, the First SPLITTER rule approach will be modified but still being observed.

   B) When the TOR option "StrictNodes" is disabled and the TOR algorithm is not able to find a route and generate a TOR circuit using the current random combination of the ENTRY NODE, MIDDLE NODE, and EXIT NODE.[27] Under this circumstance, TOR algorithm can select a TOR node from the TOR "ExcludeNodes"[27] to compose the circuit and provide a valid route to the destination.

2. The SPLITTER starts the new TOR INSTANCE. This instance will create the TOR circuits always observing the first SPLITTER rule, according to RELAY ENFORCE MODE selected and others specific TOR options.[27]

3. The SPLITTER creates a random disturb in the interval of TOR circuit creation, aiming to avoid a natural time pattern in the systematic loop process of creation and utilization of TOR circuits.

4. When the instance lifetime, reach the time limit specified by the user, the SPLITTER kills the running process related with this TOR instance, delete the temporary and all configuration files related with it and restart the life circle.

SPLITTER TOR instance life circle overview:

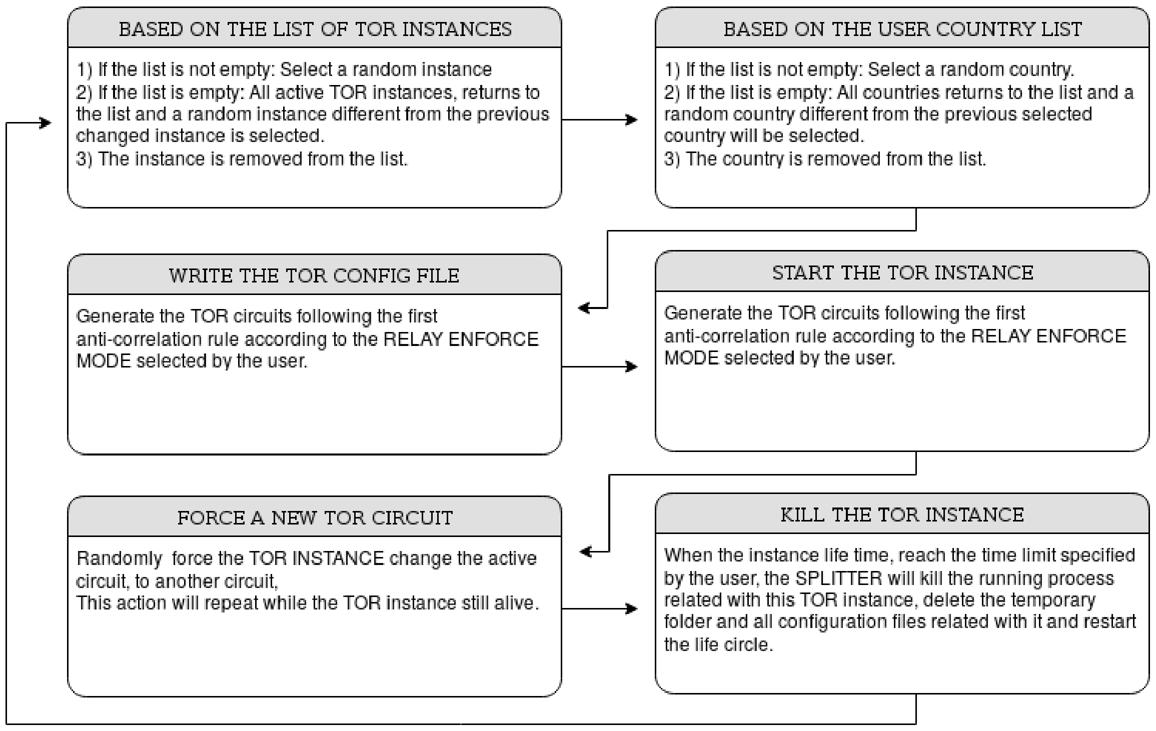

BASED ON THE LIST OF TOR INSTANCES
1) If the list is not empty: Select a random instance
2) If the list is empty: All active TOR instances, returns to the list and a random instance different from the previous changed instance is selected.
3) The instance is removed from the list.
BASED ON THE USER COUNTRY LIST
1) If the list is not empty: Select a random country.
2) If the list is empty: All countries returns to the list and a random country different from the previous selected country will be selected.
3) The country is removed from the list.
WRITE THE TOR CONFIG FILE
Generate the TOR circuits following the first anti-correlation rule according to the RELAY ENFORCE MODE selected by the user.
START THE TOR INSTANCE
Generate the TOR circuits following the first anti-correlation rule according to the RELAY ENFORCE MODE selected by the user.
FORCE A NEW TOR CIRCUIT
Randomly force the TOR INSTANCE change the active circuit, to another circuit,
This action will repeat while the TOR instance still alive.
KILL THE TOR INSTANCE
When the instance life time, reach the time limit specified by the user, the SPLITTER will kill the running process related with this TOR instance, delete the temporary folder and all configuration files related with it and restart the life circle.

Image 12: A TOR instance life circle

The total amount of simultaneous active TOR instances is calculated using:

*(X * Y) = Total amount of simultaneous active TOR instances.*

Where “*X*” is the number of countries and “*Y*” the number of desired instances inside the same country.

User case:

The options for the TOR NODE/RELAY enforcing are:

- **ENTRY:**

  Sets a specific country as ENTRY NODE and will use a different country as EXIT relay. This mode provides the best security for the user and it’s considered the default enforcing mode inside the context of SPLITTER solution.[27]
  The load balancing algorithm for HAPROXY in this mode is Round Robin.[36]
  Considering a specific country is enforced for TOR ENTRY node, the SPLITTER will select another random country from the list of countries defined by the user as TOR EXIT node, but never the same country already defined to be used as TOR ENTRY node.
  By enforcing this rule, the SPLITTER is controlling the TOR algorithm and its free random selection of countries which will compose the TOR circuit. [8, 27, Images 2 and 3]

- **EXIT:**

  Sets a specific country as EXIT NODE and will use a different country as ENTRY relay. This option gives the user the control of the EXIT relays and could be used to bypass GeoIP protections.[29]
  For example, this option is very suitable when you need to make sure that each request will hit the destination through a different country or the same country, depending on the number of countries each TOR instance can use.
  The load balancing algorithm for HAPROXY in this mode is Round Robin.[36]

A more specific SPLITTER EXIT mode user case:

A) **To always hit the target with the same country:** The user needs to include only the desired country in the list of countries available and set the number of simultaneous countries as 1 (one). The number of instances "Y" inside this same country should be adjusted according to the user's stability and speed needs.

B) **To hit the target using random countries:** After defining the list of countries SPLITTER can use, the user should define the number of simultaneous countries as "X" and the number of instances inside each country as "Y" according to his stability and speed needs.

- **SPEED:**

  This option will enforce the TOR INSTANCE use the same country as ENTRY NODE, MIDDLE NODE, and EXIT NODE. The idea is to ensure the best transmission performance of a TOR circuit, considering the restricted geolocation area that packets should travel to cross the entire TOR circuit.[8]

A short discussion about the SPLITTER SPEED MODE:

This mode could be considered unsafe due to the fact that ENTRY NODE and EXIT NODE will be selected inside the same country.

If considered the legal implications and legal bureaucracy which could variate depending on jurisdictional rules and boundaries in the perspective of digital sovereignty. Could be easier to the same adversary compromise a high number of TOR NODES inside the same country.

For example, following this proposed context, the effort necessary for a university receive the legal approval for its students conduct experiments and researches in a compromised offshore TOR NODE, could be lower if compared with the effort necessary to receive the legal authorization to do the same inside the same country. The adversary in this context could be forced to compromise a high number of TOR NODES inside the same country to conduct his experiments. The discussion behind this legal implication and bureaucracy is beyond the scope of this research. However, the feasibility of this proposed scenario affects the adversary results because do not reflect the global perspective of TOR network and make the SPLITTER speed mode a very insecure option due to the high probability of selecting a compromised TOR circuit which belongs to the same adversary. [29, 30, 31, 38, 39, 40]

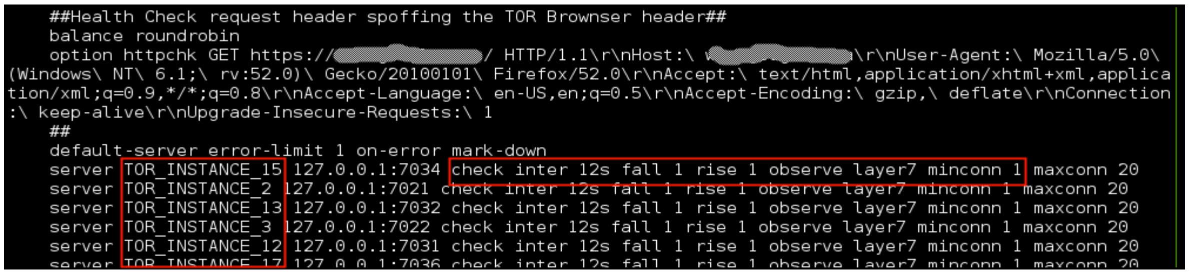
<u>How HAPROXY is used to improve the stability and performance of TOR network?</u>

In general, the stability and performance of many circuits from TOR network are not enough for High Definition media consuming like videos in 1080p for example. The TOR performance and stability related issues can compromise the user experience when trying to consume High Definition media over TOR.

The paper "Improving Tor using a TCP-over-DTLS Tunnel" from Joel Reardon and Ian Goldberg, provide a deep analysis of the TOR network performance and stability related issues. [42]

The SPLITTER is using HAPROXY to perform a health-check of the established TOR circuit before sending the user data. The user can specify a specific website and the interval of analyses. The HAPROXY will monitor the availability of the TOR circuit based on the HTTP answer. If the circuit does not answer or if the TOR EXIT node from the current circuit is not able to resolve the requested address the TOR instance is considered down. The requests are forwarded to another TOR instance until a fast, stable and reliable circuit be created in the previous instance considered down. The user can specify how many errors are necessary to consider one TOR instance as down and how many successful requests are necessary to consider it up again.

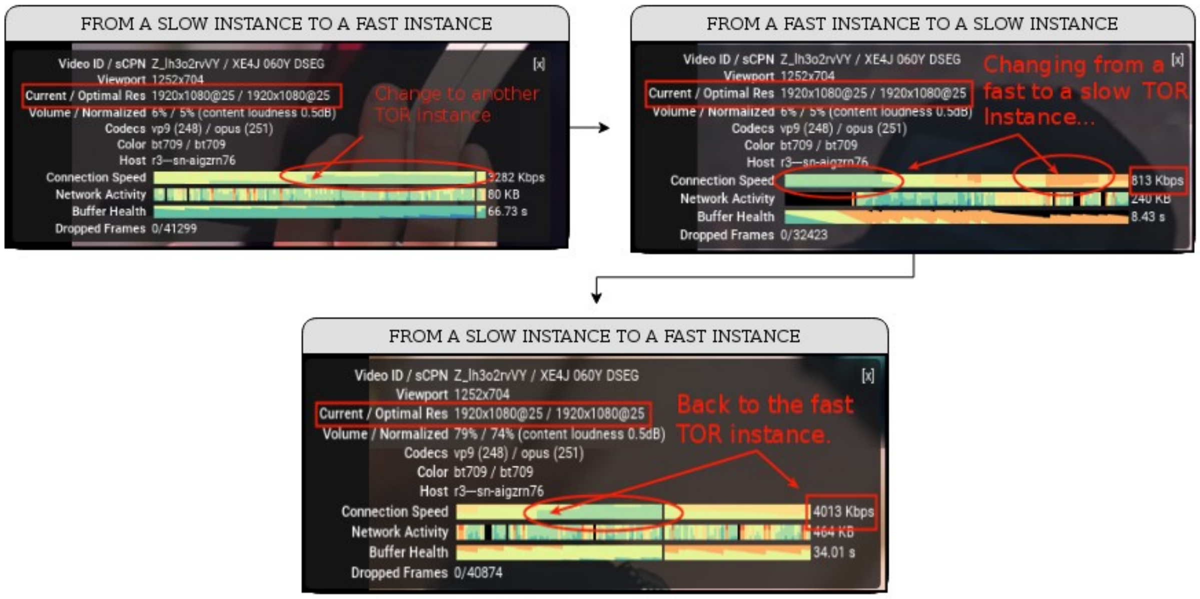
To difficult the correlation, the SPLITTER is using a random order of TOR instances inside HAPROXY configuration file. The idea is to avoid that consecutive requests being sent through the same country when the users decide to use more than one tor instance per country. The interval between the checks can be random or fixed, the user will adjust it according to the speed which new TOR circuits are being and destroyed. By default, 12s is used as the fixed interval between the health-checks. However, we should consider that the health-check by its self can generate a pattern and the adversary can use this sequence of checks to track the user. In another hand, the health-check provide a better speed and stability allowing the user consume High Definition movies even using the TOR network.



```
    ##Health Check request header spoffing the TOR Brownser header##
    balance roundrobin
    option httpchk GET https://[redacted]/ HTTP/1.1\r\nHost:\ [redacted]\r\nUser-Agent:\ Mozilla/5.0\
(Windows\ NT\ 6.1;\ rv:52.0)\ Gecko/20100101\ Firefox/52.0\r\nAccept:\ text/html,application/xhtml+xml,applica
tion/xml;q=0.9,*/*;q=0.8\r\nAccept-Language:\ en-US,en;q=0.5\r\nAccept-Encoding:\ gzip,\ deflate\r\nConnection
:\ keep-alive\r\nUpgrade-Insecure-Requests:\ 1
    ##
    default-server error-limit 1 on-error mark-down
    server TOR_INSTANCE_15 127.0.0.1:7034 check inter 12s fall 1 rise 1 observe layer7 minconn 1 maxconn 20
    server TOR_INSTANCE_2 127.0.0.1:7021 check inter 12s fall 1 rise 1 observe layer7 minconn 1 maxconn 20
    server TOR_INSTANCE_13 127.0.0.1:7032 check inter 12s fall 1 rise 1 observe layer7 minconn 1 maxconn 20
    server TOR_INSTANCE_3 127.0.0.1:7022 check inter 12s fall 1 rise 1 observe layer7 minconn 1 maxconn 20
    server TOR_INSTANCE_12 127.0.0.1:7031 check inter 12s fall 1 rise 1 observe layer7 minconn 1 maxconn 20
    server TOR_INSTANCE_17 127.0.0.1:7036 check inter 12s fall 1 rise 1 observe layer7 minconn 1 maxconn 20
```

Image 13: A screen shot from HAPROXY configuration file, which show the Health Check configuration.

The difference between the performance and stability from different active TOR instance can be observed by analyzing the sudden drop or increase of connection speed. Even a user without great experience with network monitoring can observe this behavior using the feature “Stats for nerds” provided by Youtube.com. When the TCP stream is transferred from one instance to another is possible to observe a very fast changing in the connection speed. The sequence of screenshots below shows this behavior.


FROM A SLOW INSTANCE TO A FAST INSTANCE
Change to another TOR Instance
FROM A FAST INSTANCE TO A SLOW INSTANCE
Changing from a fast to a slow TOR Instance...
FROM A SLOW INSTANCE TO A FAST INSTANCE
Back to the fast TOR instance.

```
   Video ID / sCPN Z_lh3o2rvVY / XE4J 060Y DSEG
          Viewport 1252x704
Current / Optimal Res 1920x1080@25 / 1920x1080@25
Volume / Normalized 6% / 5% (content loudness 0.5dB)
            Codecs vp9 (248) / opus (251)
             Color bt709 / bt709
              Host r3—sn-aigzrn76
  Connection Speed 3282 Kbps
  Network Activity 80 KB
     Buffer Health 66.73 s
    Dropped Frames 0/41299
```

```
   Video ID / sCPN Z_lh3o2rvVY / XE4J 060Y DSEG
          Viewport 1252x704
Current / Optimal Res 1920x1080@25 / 1920x1080@25
Volume / Normalized 6% / 5% (content loudness 0.5dB)
            Codecs vp9 (248) / opus (251)
             Color bt709 / bt709
              Host r3—sn-aigzrn76
  Connection Speed 813 Kbps
  Network Activity 240 KB
     Buffer Health 8.43 s
    Dropped Frames 0/32423
```

```
   Video ID / sCPN Z_lh3o2rvVY / XE4J 060Y DSEG
          Viewport 1252x704
Current / Optimal Res 1920x1080@25 / 1920x1080@25
Volume / Normalized 79% / 74% (content loudness 0.5dB)
            Codecs vp9 (248) / opus (251)
             Color bt709 / bt709
              Host r3—sn-aigzrn76
  Connection Speed 4013 Kbps
  Network Activity 464 KB
     Buffer Health 34.01 s
    Dropped Frames 0/40874
```

Image 14: Screenshots, showing the connection speed dropping or speeding up according to the performance of TOR instance after change

SPLITTER NETWORK

To difficult the natural correlation between the TOR network and the user, we mentioned the need to connect in a public VPN service before connecting in TOR network as demonstrated in the image 9. This simple approach prevents that the internet provider is able to see that user is connected on TOR network. The encryption of TOR network prevents that the VPN provider is able to see what user is doing inside TOR network as demonstrated in the images 2 and 3. This is considered the best approach for privacy because the natural correlation between the TOR network and the user has been broke.[38, 39, 40]

However, more sophisticated traffic analyses techniques can observe the natural traffic patterns and follow the path from the VPN provider until the user.[1]

To difficult even more this possibility of correlation, a low-cost VPS and VPN network should be considered.[38, 39, 40]

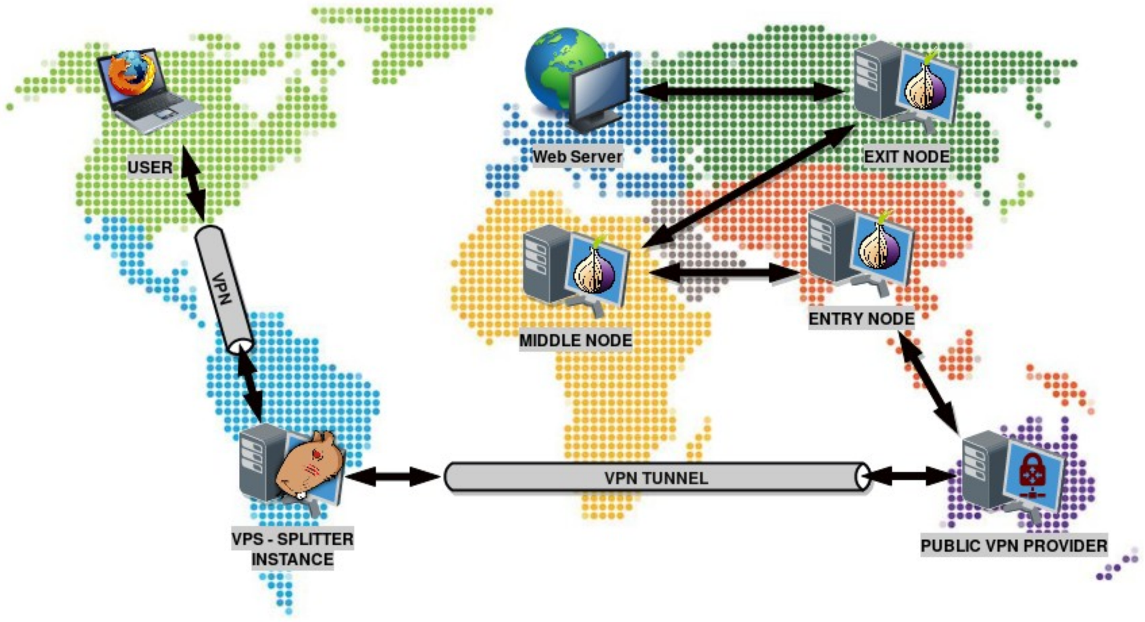


Image 15: A global representation about the TCP packet path inside the SPLITTER NETWORK

1. **The VPS will act as VPN SERVER and VPN CLIENT at the same time:**

   - The user will connect to the VPS using a VPN service running in the VPS. This VPN assume the default gateway for the user machine and all data from the user machine will be forwarded to VPS. The user should point his browser to the exposed HAPROXY port. More details in following number 3.

   - The VPS is also connected in a **PUBLIC VPN** service and all output traffic from the VPS is send using the public VPN connection. This way, when the VPS execute the SPLITTER script the TOR network connection will be established using the public VPN as demonstrated in the image 9. [38, 39, 40]

2. **The VPS have a firewall to avoid leaks:**

   The inbound traffic and the outbound traffic is controlled to avoid leaks and enforce the following:

   - The only inbound traffic allowed is to VPN SERVER service running in the VPS.
     All others input traffic are blocked including inputs from docker network.

- The outbound traffic allowed is to resolve the DNS address from the PUBLIC VPN server, to connect to this public VPN service and to allow the VPS to connect into HAPROXY port exposed by the docker container.
  All others output traffic is blocked including traffic from the user connected in the VPN SERVER to the internet. The only output route for the user connected to the VPN SERVER is using the TOR connection running inside the docker container.

3. **<u>The VPS will execute a docker container with the SPLITTER solution:</u>**

The SPLITTER solution will be executed inside a Docker container, and the connection with the public VPN service will be created in the VPS operating system and transferred to the docker container. It is necessary because the public VPN connection should assume the default gateway of the Docker container, but can not assume the default gateway from the VPS operating system.

With this docker container approach, we will ensure that the TOR network connection will be executed inside the container and use the public VPN service. The Docker container will expose only the HAPROXY port to the VPS and the user should connect on the VPS VPN SERVICE and point his browser to the HAPROXY port exposed to the VPS.[44, 45]

The container will provide an extra security layer. If the adversary is able to exploit any vulnerability and assume the control of the container, he is trapped inside the container context and the only output available is using the public VPN service. Following this scenario, the attacker will face more difficult to interact with the operating system of the VPS and interact with the user connected in the VPS due to the segregation of environments provided by Docker.[44, 45]

<u>The SPLITTER network in a global scale:</u>

This low-cost network can be distributed around the globe using different VPS providers and different VPN providers. Each VPS will execute a docker container with the SPLITTER SOLUTION running inside, connected to a different VPN provider before connect to the TOR network, according demonstrated in the images 9, 10, 11 and 15.

Each VPS should have at least 1GB of memory RAM and will cost the average of $200,00 (two hundred dollars) per year to remain online.

Usually, the public VPN plans will allow the client to have at least 3 simultaneous connections and the price for 1 year is $100,00 (one hundred dollars).

This scenario allows the user to create his own private combination of VPS, VPN, and TOR. The user can use the HAPROXY once again to perform the load balancing between all VPS running the SPLITTER solution. The result will be an even better global spread of the traffic, difficulting the correlation between the TOR network and the final user.

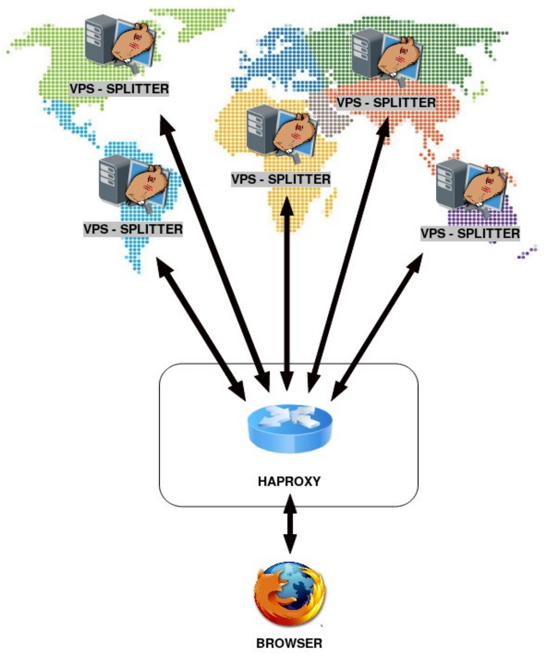


Image 16: A global scale representation about the SPLITTER NETWORK

VPS iptables firewall bash script sample:

```
#!/bin/bash
echo "
*filter
   #Drop everything by default
   :INPUT DROP [0:0]
   :FORWARD DROP [0:0]
   :OUTPUT DROP [0:0]

   #Allow Loopback
   -A INPUT -i lo -j ACCEPT
   -A OUTPUT -o lo -j ACCEPT

   #Allowing only related/established traffic
   -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
   -A OUTPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

   #Allow Output to Docker
   -A OUTPUT -o docker+ -j ACCEPT

   #Block Docker input
   -A INPUT -i docker+ -j DROP

   #Allow the user connect in VPN Service running at port 443
   -A INPUT -i eth0 -p tcp --dport 443 -j ACCEPT

   # Allow the input traffic comming from the VPN client to VPS
   -A INPUT -i as0t0 -j ACCEPT

   # Allow the output traffic from the VPS to the VPN client
   -A OUTPUT -o as0t0 -j ACCEPT

   #Allow only output for the VPN server port 1195
   -A OUTPUT -o eth0 -p udp --dport 1195 -j ACCEPT
   -A OUTPUT -o eth0 -p udp -d 8.8.8.8 --dport 53 -j ACCEPT

   ##### Drop all others #######
   -A INPUT -j DROP
   -A FORWARD -j DROP
   -A OUTPUT -j DROP
COMMIT

*nat
   -A POSTROUTING -o eth0 -j MASQUERADE
   -A PREROUTING -i as0t0 -p tcp --dport 63536 -j REDIRECT --to-port 63536 -m comment
   --comment "REDIRECT VPN CLIENT to exposed HAPROXY proxy port"
   -A PREROUTING -i as0t0 -p tcp --dport 63537 -j REDIRECT --to-port 63537 -m comment
   --comment "REDIRECT VPN CLIENT to exposed HAPROXY Status port"
COMMIT" > ipv4_firewall_rules.cfg
iptables-restore ipv4_firewall_rules.cfg
echo "
*filter
   :INPUT DROP [0:0]
   :FORWARD DROP [0:0]
   :OUTPUT DROP [0:0]
COMMIT" > ipv6_firewall_rules.cfg
ip6tables-restore ipv6_firewall_rules.cfg
```

VPS docker and VPN bash script sample:

```
#!/bin/bash
splitter_docker_img_id="$(docker image ls -a | grep -i "splitter" | awk '{print $3}')"
linux_low_privilleged_user_id="splitter"
echo "Stopping all process related with OPENVPN, PRIVOXY, TOR and HAPROXY."
   killall openvpn
   killall privoxy
   killall tor
   killall haproxy
   docker ps -a | awk '{print $1}' | grep -iv 'CONTAINER' > docker_containers_list.txt
   while read container_id
   do
      docker stop ${container_id} 2>&1 > /dev/null
      docker container kill ${container_id} 2>&1 > /dev/null
      docker container rm -f ${container_id} 2>&1 > /dev/null
   done < docker_containers_list.txt
   rm docker_containers_list.txt
   service docker stop

#Restoring the firewall rules
   iptables-restore ipv4_firewall_rules.cfg
   ip6tables-restore ipv6_firewall_rules.cfg

#Start docker server and container
   service docker start
   sleep 3
   docker run -d -i -p 63536:63536 -p 63537:63537 --name splitter_v.0.0.1 $
{splitter_docker_img_id} /bin/bash
   sleep 4

#Mapping the docker network namespace inside the 'ip netns' command context
   CONTAINERID="$(docker ps -a | awk '{print $1}' | grep -iv "CONTAINER")"
   CONTAINERPID="$(docker inspect -f '{{.State.Pid}}' ${CONTAINERID})"
   rm -rf /var/run/netns/*
   mkdir -p /var/run/netns
   ln -sf /proc/${CONTAINERPID}/ns/net "/var/run/netns/${CONTAINERID}"

#Connecting to the PUBLIC VPN service using a low-privilleged account
   /usr/sbin/openvpn --user ${linux_low_privilleged_user_id} --pull-filter ignore
redirect-gateway --config YOUR_PUBLIC_VPN_OPENVPN_PROFILE.ovpn&
   sleep 5

#Enable Kernel Forward packets
   echo 1 > /proc/sys/net/ipv4/ip_forward

#Transfering the PUBLIC VPN connection to the docker container and setting it as default
gateway for the container.
   IFVPN="$(ifconfig -a | grep "tun" | cut -d ":" -f 1)"
   IPVPN="$(ifconfig ${IFVPN} | grep "inet " | awk '{print $2}')"
   ip link set ${IFVPN} netns ${CONTAINERID}
   ip netns exec ${CONTAINERID} ip addr add ${IPVPN}/32 dev ${IFVPN}
   ip netns exec ${CONTAINERID} ip link set ${IFVPN} up
   ip netns exec ${CONTAINERID} route del default
   ip netns exec ${CONTAINERID} route add default ${IFVPN}

#Executing the SPLITTER solution inside the SPLITTER docker container
   docker exec -t -i ${CONTAINERID} /bin/bash -i -c "/bin/bash
start_splitter.sh;/bin/bash"
```

SPLITTER PROOF OF CONCEPT

Despite all my limitations as independent researcher like for example the absence of compromised TOR nodes in the real world scenario to try to reproduce every single de-anonymization technique referenced in this paper and check how effective the SPLITTER approach will be against each technique. A simple but effective sequence of tests was executed and the initial results show an impressive result even considering all the limitations.

The main propose behind SPLITTER is difficult the correlation and provide a better user experience while using the TOR network. As mentioned before, the total amount of data that adversary can collect could directly affect the final result of de-anonymization techniques and also the ability of this adversary to analyze, correlate and make decisions based on collected data. [1, 2, 3, 4, 5, 6, 10, 20, 23, 28]

From the perspective of de-anonymization techniques the most important observed points were:

- The time of interaction between the user and the supposed compromised node. Each de-anonymization technique shows a different amount of time, which is considered necessary to observe the injected or the natural pattern and exclude the false-positive. I decide to use **almost 7x times** the period which user will be exposed or interact with a supposed compromised TOR node to cover almost all possibility of time necessary for each de-anonymization technique. The time used during my tests was **34 minutes and 14 seconds**.
- For example, Sambuddho Chakravarty, Marco V. Barbera, Georgios Portokalidis, Michalis Polychronakis and Angelos D. Keromytis mentioned in the paper “On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records” mentioned that was necessary **at least 5 – 7 minutes** to generated sustained traffic to identify the pattern.[1] Also, we should remember that using this technique they have 81% of accuracy in the identification of users in the real world scenario. How tolerant will this kind of technique be against the global distribution of traffic provided by SPLITTER? [Image 5]

- How effective similar techniques which intercept and inject malicious code driving the browser to send consecutive requests to a controlled malicious web server will remain reliable, considering that now a new geographical path will be used for each HTTP request. How will they be able to still analyzing the interval of requests? A similar technique has been demonstrated by Nathan S. Evans, Roger Dingledine, and Christian Grothoff in the paper - “A Practical Congestion Attack on Tor Using Long Paths”. For this technique, the correlation between the time of each request received by the malicious web server or the compromised TOR NODES were crucial for the results.[2] How can the TOR load balancing help the victim escape from similar techniques which rely on the ability of the adversary to compromise the stability and performance from a specific TOR NODE/RELAY?

- The last and maybe the most important questions for the tests are: After applying the SPLITTER approach, what is the total amount of data that the adversary will have to analyze and correlate for each compromised TOR NODE / RELAY under his control? I will consider the worst case scenario where the adversary was able to compromise all NODES/RELAY that SPLITTER use to reach the target. What is the effort necessary to correlate everything?

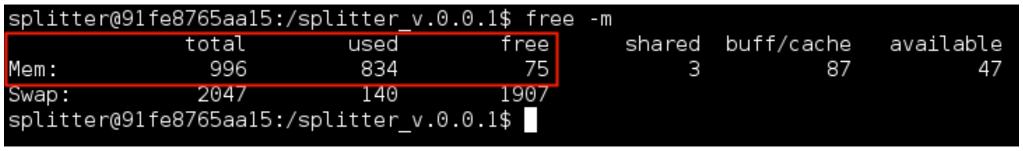
The setup of test environment:

About the list of selected countries:

As mentioned the SPLITTER approach is based in the geolocation of ENTRY NODES and EXIT NODES. The total amount of countries selected and defined as the default for SPLITTER are 25 countries and this number is based on the number of available EXIT NODES and ENTRY NODES for each selected country. In this case, the country should have at least 3 EXIT NODES and at least 3 ENTRY GUARDS. By default, the SPLITTER is using only the ENTRY GUARDS because they could be considered more secure.[21, 22, 24] This decision reduce the list of available ENTRY NODES but could be considered a better approach from the perspective of security.[16]

About the number of simultaneous active Countries/Instances:

Many tests have been performed with a different number of simultaneous active TOR instances/countries. However, I decided to use the total amount of 20 simultaneous active countries/instances as default for the SPLITTER. This number is based on the total amount of memory RAM necessary to execute at least 20 simultaneous TOR instances. The SPLITTER will consume the average of 800Mb **-** 900Mb of RAM to keep 20 instances running.



```
splitter@91fe8765aa15:/splitter_v.0.0.1$ free -m
              total        used        free      shared  buff/cache   available
Mem:            996         834          75           3          87          47
Swap:          2047         140        1907
splitter@91fe8765aa15:/splitter_v.0.0.1$ 
```

Image 17: A screenshot from the low-cost VPS showing the amount of memory used.

The reason why was decided to keep the average of memory RAM consuming under 1GB is related to the price of a low-cost VPS with at least 1GB of RAM. This way is easy to distribute and create a global distribution of SPLITTER, but this topic will be covered later. For the results presented in this paper comprehends only one machine running the SPLITTER.

About the country rotation interval between the simultaneous active TOR instances:

In order to simulate a scenario where the user is trying to evade the de-anonymization technique with the better average of successful identification of users behind the TOR network in the real world scenario which is 81.4% with an average of false positive rate of 6.4%.[1] I decided to set up the maximum lifetime for a TOR instance as **4 minutes**. This decision was made based on the minimum average of time necessary for a TCP stream in this de-anonymization technique which is **5 minutes**.[1]

To keep a minimum of stability and performance necessary to watch a video em High Definition 720p I decided to destroy and recreate **10 simultaneous instances** after every **2 minutes**. This way we have a good balance between stability, performance, and security for the user using this SPLITTER.

With this setup, we have the minimum lifetime for a TCP stream set as **2 minutes** and the maximum set as **4 minutes**.

<u>About how the data has been collected:</u>

Two different ways to collect data were applied due to my limitation as an independent researcher and the absence of a sustainable number of compromised TOR NODES under my control.

1. First, the total amount of **10.000 consecutive HTTP requests** were sent to websites hosting API's designed to show the IP address of the request. Two websites were used outside of my lab for a better proximity with the reality. During the scenario of tests, I'm considering that these web servers are under the control of the adversary and can inject a malicious pattern in my traffic.
   A) http://ipinfo.io/ip
   B) https://api.ipify.org/?format=text

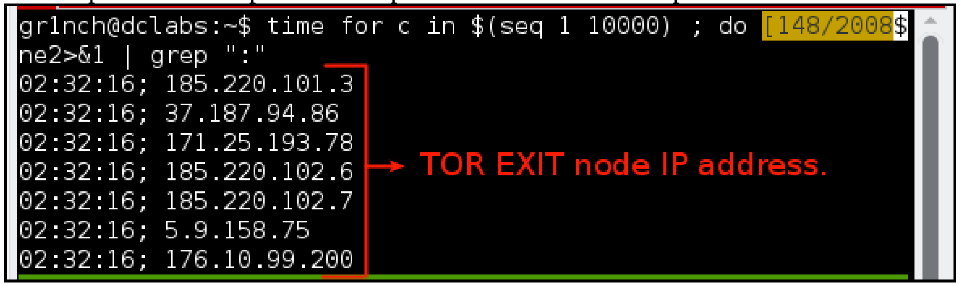
This way the IP of EXIT NODE used to hit the target can be observed. Considering for this scenario that all EXIT NODES used to hit the target are compromised and try to evaluate how many data the adversary was able to collect with the SPEED MODE ENFORCING and later with the EXIT NODE ENFORCING was also used. The result of both enforcing relay method is presented as one due to the fact that my limitations prevent me to see the ENTRY NODE used and the same average of consecutive hits from the same EXIT NODE in both methods.

Later the same test was repeated but considering only **1.000 requests** and **1 request per second**, only to simulate a normal web surfing without download.

To accomplish this task a simple shell script function was executed in a loop until complete the proposed number of HTTP requests:

```
#!/bin/bash
show_ip_function() {
echo "$(echo -n "$(date | awk '{print $4}'); " && echo
"$(curl http://ipinfo.io/ip 2> /dev/null)")"&}
time for c in $(seq 1 10000) ; do show_ip_function
;done 2> /dev/null
```

The output of this simple shell script is similar with the sample below:



```
gr1nch@dclabs:~$ time for c in $(seq 1 10000) ; do [148/2008$
ne2>&1 | grep ":"
02:32:16; 185.220.101.3
02:32:16; 37.187.94.86
02:32:16; 171.25.193.78
02:32:16; 185.220.102.6
02:32:16; 185.220.102.7
02:32:16; 5.9.158.75
02:32:16; 176.10.99.200
```

Image 18: A screenshot showing the IP of the EXIT NODE used to hit the target.

2. The download of a huge file from my controlled web server. It was necessary to use a file huge enough to keep the download for the same 34 minutes and 14 seconds used to complete the 10.000 HTTP requests. To reduces the chances of interference from malwares or scanners running in the wild internet, the port 63537 has been used in the web server.

To simulate the scenario where the adversary was able to intercept the ENTRY NODE and EXIT NODE. I executed 2 tests using the SPLITTER SPEED MODE.

More 2 tests were executed using the SPLITTER EXIT MODE to simulate the scenario where the adversary was able to compromise only the EXIT NODE.

The same average of consecutive utilization of a supposed compromised EXIT NODE has been observed during all tests. Due to the similarity of the results only the last test is detailed in the following pages.

<u>In summary the first 4 tests were based on:</u>

Available countries	32
Simultaneous active Countries / Instances	20
Simultaneous countries changed during the country rotation	10
Interval between the country rotation	2 minutes
Concurrent requests per second.	5~20
Time of exposition or interaction between the user and the supposed compromised web server.	34 minutes and 14 seconds

The results:

Summary:

With the proposed SPLITTER configuration, the first 4 tests confirm a very similar result. The worst case scenario an adversary which controls only 1(one) compromised TOR EXIT NODE will be able to intercept only 3.6% percent of total amount of data sent by the user through TOR network. In the best scenario, only 0.5% will be intercepted. In order to have the same amount of data necessary for the de-anonymization techniques, the adversary will need to compromise all TOR EXIT NODES and be able to correlate the data intercepted from all compromised TOR EXIT NODE.

Details:

For 34 minutes and 14 seconds, 10.000 requests has been sent using 413 different TOR EXIT NODES.
The result shows that in the worst case, the maximum amount of data that one adversary would be able to intercept for each compromised TOR EXIT NODE is 3.6% from the total of requests sent by the victim using the SPLITTER approach. All others EXIT NODES has been used under this value and the average is 0.5% for each compromised EXIT NODE.

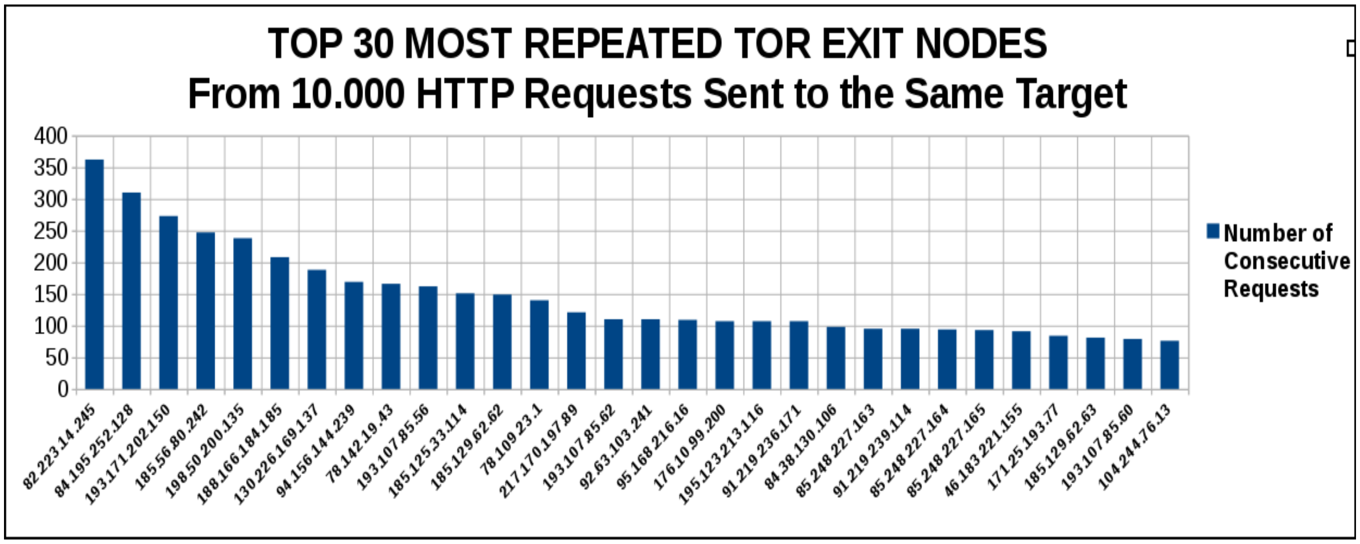


Image 19: The overview about the most repeated TOR EXIT NODES

An overview about how the 10.000 HTTP requests have been spread between the supposed 413 compromised TOR EXIT NODES show that SPLITTER was able to keep the overall utilization of repeated EXIT NODES under 0.5%.

In the context of the first 4 tests, it means that 50 requests from 10.000 have being sent through each compromised EXIT NODE.

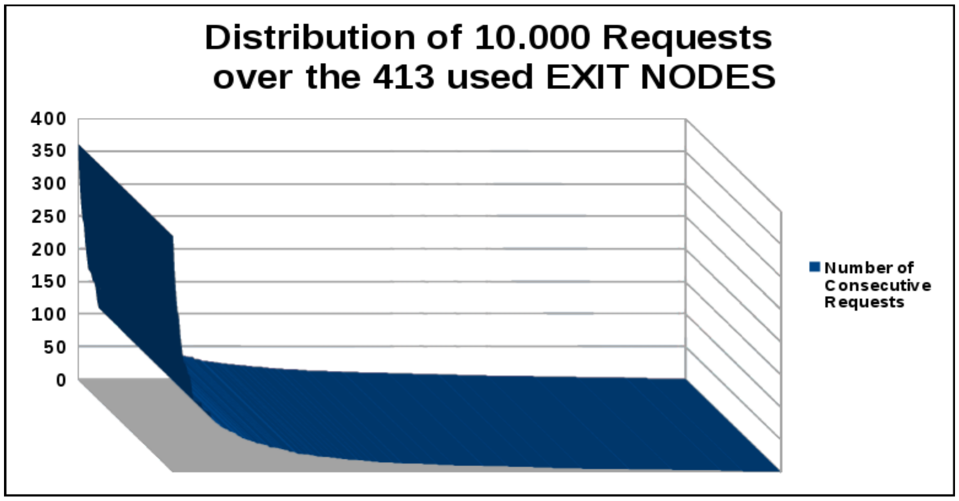


Image 20: The overview about the most repeated TOR EXIT NODES

Considering that the same adversary controls the 7 most used EXIT NODES, he will be able to collect only 17.9% of the total amount of data transferred by the victim.

FROM 10.000 REQUESTS SENT TO THE SAME WEB SERVER			
EXIT NODE	**Country**	**CONSECUTIVE REQUESTS**	**PERCENT**
82.223.14.245	Spain (ES)	362	3.6%
84.195.252.128	Belgium (BE)	310	3.1%
193.171.202.150	Austria (AT)	273	2.7%
185.56.80.242	Seychelles (SC)	247	2.4%
198.50.200.135	Panama (PA)	238	2.3%
188.166.184.185	Netherlands (NL)	208	2.8%
130.226.169.137	Denmark (DK)	188	1.8%
Total amount of data collected by this adversary			**17.9%**

An analyze about the interval between the consecutive requests hitting the supposed compromised web server through the same supposed compromised EXIT NODE, show that SPLITTER was able to create a random disturb in the time. This random disturbs can difficult the time correlation and compromise the pattern injected by the adversary.

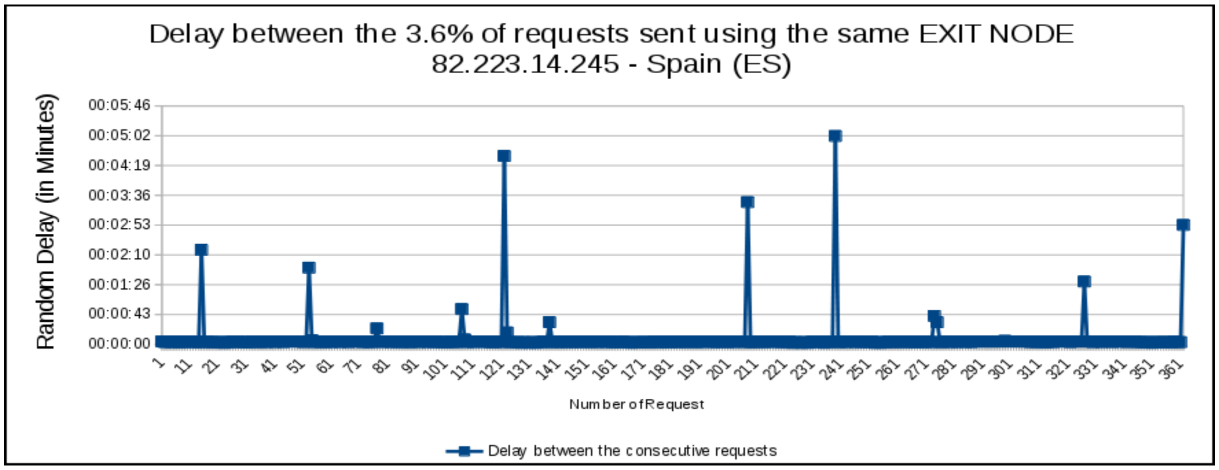
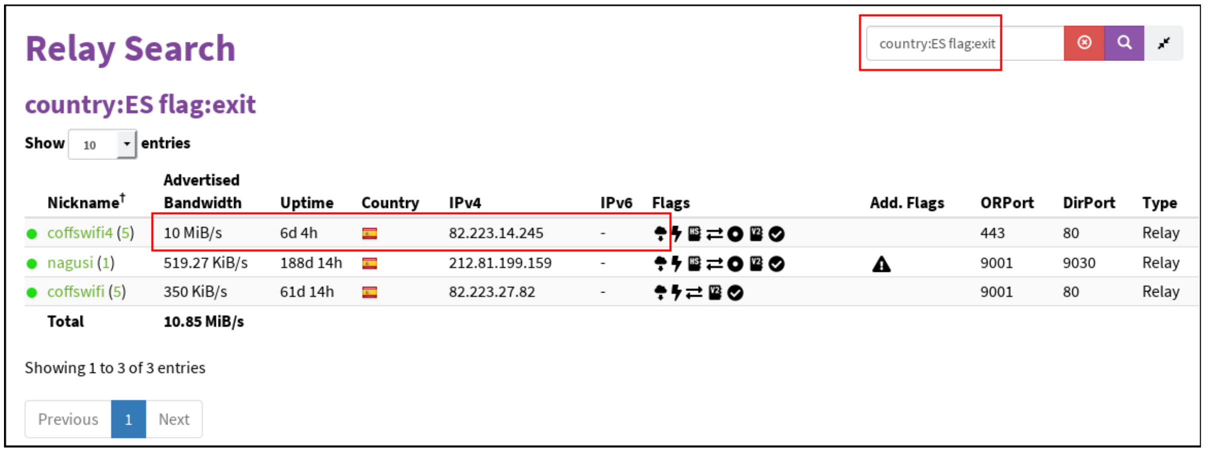


Image 21: A representation of time disturbs which can compromised the time correlation and difficult the identification of pattern.

Investigating the reason why we have the overall 0.5% and only 3.6% for some specific countries I discover that according to the official TOR METRICS web page. By the time of this research, SPAIN (ES) has only 3 EXIT NODES with math with my initial countries requirement, but only 1 has bandwidth enough to support the minimum average of 5 concurrent requests. This impacted the previously presented results and explains why we have the notorious difference in the graphics.



Relay Search

country:ES flag:exit

country:ES flag:exit

Show 10 entries

Nickname†	Advertised Bandwidth	Uptime	Country	IPv4	IPv6	Flags	Add. Flags	ORPort	DirPort	Type
● coffswifi4 (5)	10 MiB/s	6d 4h	🇪🇸	82.223.14.245	-	ϟ ⇄ ● ✓		443	80	Relay
● nagusi (1)	519.27 KiB/s	188d 14h	🇪🇸	212.81.199.159	-	ϟ ⇄ ● ✓	⚠	9001	9030	Relay
● coffswifi (5)	350 KiB/s	61d 14h	🇪🇸	82.223.27.82	-	ϟ ⇄ ✓		9001	80	Relay
Total	**10.85 MiB/s**									

Showing 1 to 3 of 3 entries

Previous 1 Next

Image 22: A screen shot from https://metrics.torproject.org/rs.html#search/country:ES%20flag:exit [43]

A similar situation was observed in other countries which presented a high number of consecutive requests sent through the same EXIT NODE. These results show the importance of always select countries with a sustainable number of ENTRY GUARDS and EXIT NODES. The user can perform tests as presented here to help him select the better combination of countries reducing, even more, the total amount of data prone to be collected by the same compromised TOR relay.

Results related with stability and performance:

The results show that the total amount of lost requests is under 1% considering that user sent 10.000 HTTP requests to the same target. By using HAPROXY, the SPLITTER solution was able to guarantee a very acceptable level or stability. The global load balancing keeps the average of 10 consecutive requests per second during the time where the lowest rate of consecutive requests was 5 concurrent request per second and the maximum was 20 concurrent requests per second sent to the destination target.

The number of 20 concurrent requests were reached only during the download tests.

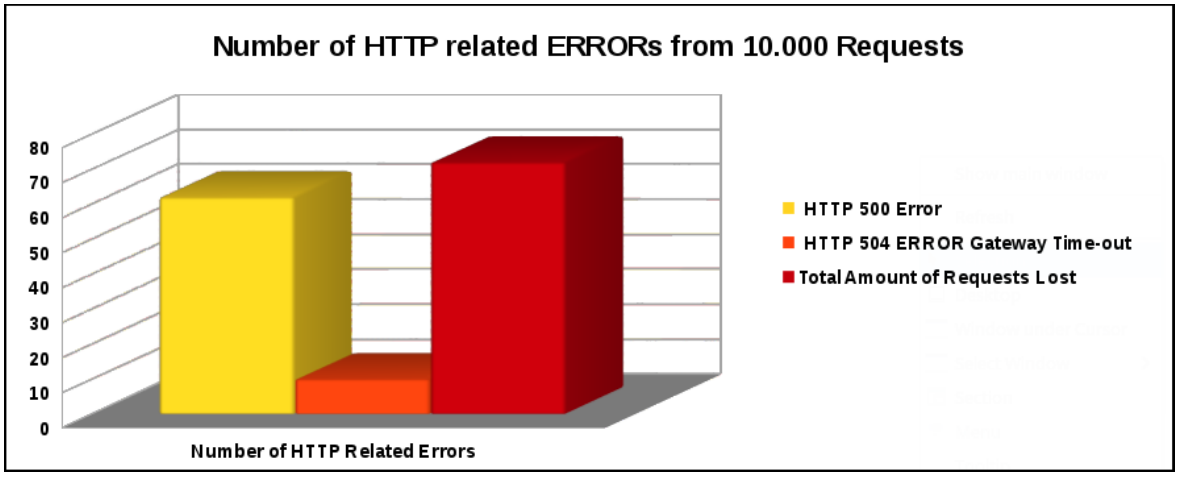
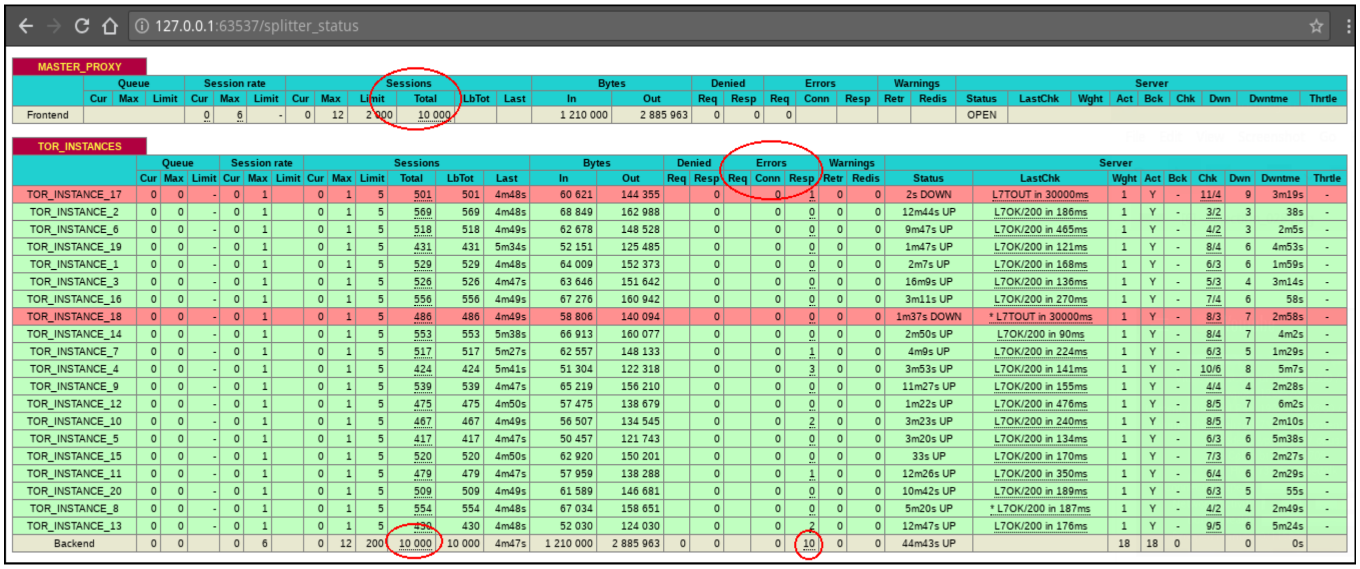


Image 23: An overview about the stability provided by HAPROXY

Considering that only 20 TOR instances have been executed during this test, I can affirm based on the HAPROXY status screen that below that no concurrent session has been sent using the same TOR circuit during the execution of 10.000 HTTP request to the same target. This screen also shows among many other details, the number of time that each TOR instance has been considered down. The health check interval used was the SPLITTER default 12 seconds, which guarantee at least one health for each created TOR circuit, if considering that by default we have a new TOR circuit every 10 seconds.

By the time of the screenshot, the TOR instances number 17 and 18 was considered down and was not part of the farm. They return to the HAPROXY farm when the health check returns a positive result as described in the Image 13.

Another important item to see on this screen is how the SPLITTER is using a random order to organize the TOR instances inside the HAPROXY server farm. Considering the systematic sequence of request created by the Round Robin algorithm, to put the TOR instances in the farm using a random order is necessary to avoid that consecutive requests be sent through the same country when the users decide to use more than one tor instance per country.



127.0.0.1:63537/splitter_status

MASTER_PROXY

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
Frontend				0	6	-	0	12	2 000	10 000			1 210 000	2 885 963	0	0	0					OPEN								

TOR_INSTANCES

	Queue			Session rate			Sessions						Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thrtle
TOR_INSTANCE_17	0	0	-	0	1		0	1	5	501	501	4m48s	60 621	144 355		0		0	1	0	0	2s DOWN	L7TOUT in 30000ms	1	Y	-	11/4	9	3m19s	-
TOR_INSTANCE_2	0	0	-	0	1		0	1	5	569	569	4m48s	68 849	162 988		0		0	0	0	0	12m44s UP	L7OK/200 in 186ms	1	Y	-	3/2	3	38s	-
TOR_INSTANCE_6	0	0	-	0	1		0	1	5	518	518	4m49s	62 678	148 528		0		0	0	0	0	9m47s UP	L7OK/200 in 465ms	1	Y	-	4/2	3	2m5s	-
TOR_INSTANCE_19	0	0	-	0	1		0	1	5	431	431	5m34s	52 151	125 485		0		0	0	0	0	1m47s UP	L7OK/200 in 121ms	1	Y	-	8/4	6	4m53s	-
TOR_INSTANCE_1	0	0	-	0	1		0	1	5	529	529	4m48s	64 009	152 373		0		0	0	0	0	2m7s UP	L7OK/200 in 168ms	1	Y	-	6/3	6	1m59s	-
TOR_INSTANCE_3	0	0	-	0	1		0	1	5	526	526	4m47s	63 646	151 642		0		0	0	0	0	16m9s UP	L7OK/200 in 136ms	1	Y	-	5/3	4	3m14s	-
TOR_INSTANCE_16	0	0	-	0	1		0	1	5	556	556	4m49s	67 276	160 942		0		0	0	0	0	3m11s UP	L7OK/200 in 270ms	1	Y	-	7/4	6	58s	-
TOR_INSTANCE_18	0	0	-	0	1		0	1	5	486	486	4m49s	58 806	140 094		0		0	0	0	0	1m37s DOWN	* L7TOUT in 30000ms	1	Y	-	8/3	7	2m58s	-
TOR_INSTANCE_14	0	0	-	0	1		0	1	5	553	553	5m38s	66 913	160 077		0		0	0	0	0	2m50s UP	L7OK/200 in 90ms	1	Y	-	8/4	7	4m2s	-
TOR_INSTANCE_7	0	0	-	0	1		0	1	5	517	517	5m27s	62 557	148 133		0		0	1	0	0	4m9s UP	L7OK/200 in 224ms	1	Y	-	6/3	5	1m29s	-
TOR_INSTANCE_4	0	0	-	0	1		0	1	5	424	424	5m41s	51 304	122 318		0		0	3	0	0	3m53s UP	L7OK/200 in 141ms	1	Y	-	10/6	8	5m7s	-
TOR_INSTANCE_9	0	0	-	0	1		0	1	5	539	539	4m47s	65 219	156 210		0		0	0	0	0	11m27s UP	L7OK/200 in 155ms	1	Y	-	4/4	4	2m28s	-
TOR_INSTANCE_12	0	0	-	0	1		0	1	5	475	475	4m50s	57 475	138 679		0		0	0	0	0	1m22s UP	L7OK/200 in 476ms	1	Y	-	8/5	7	6m2s	-
TOR_INSTANCE_10	0	0	-	0	1		0	1	5	467	467	4m49s	56 507	134 545		0		0	2	0	0	3m23s UP	L7OK/200 in 240ms	1	Y	-	8/5	7	2m10s	-
TOR_INSTANCE_5	0	0	-	0	1		0	1	5	417	417	4m47s	50 457	121 743		0		0	0	0	0	3m20s UP	L7OK/200 in 134ms	1	Y	-	6/3	6	5m38s	-
TOR_INSTANCE_15	0	0	-	0	1		0	1	5	520	520	4m50s	62 920	150 201		0		0	0	0	0	33s UP	L7OK/200 in 170ms	1	Y	-	7/3	6	2m27s	-
TOR_INSTANCE_11	0	0	-	0	1		0	1	5	479	479	4m47s	57 959	138 288		0		0	1	0	0	12m26s UP	L7OK/200 in 350ms	1	Y	-	6/4	6	2m29s	-
TOR_INSTANCE_20	0	0	-	0	1		0	1	5	509	509	4m49s	61 589	146 681		0		0	0	0	0	10m42s UP	L7OK/200 in 189ms	1	Y	-	6/3	5	55s	-
TOR_INSTANCE_8	0	0	-	0	1		0	1	5	554	554	4m48s	67 034	158 651		0		0	0	0	0	5m20s UP	* L7OK/200 in 187ms	1	Y	-	4/2	4	2m49s	-
TOR_INSTANCE_13	0	0	-	0	1		0	1	5	430	430	4m48s	52 030	124 030		0		0	2	0	0	12m47s UP	L7OK/200 in 176ms	1	Y	-	9/5	6	5m24s	-
Backend	0	0		0	6		0	12	200	10 000	10 000	4m47s	1 210 000	2 885 963	0	0		0	10	0	0	44m43s UP		18	18	0		0	0s	

Image 24: A screenshot about HAPROXY status when the 10.000 request were sent to the supposed compromised web server.

FINAL CONSIDERATION AND NEXT CHALLENGE

Based on the average of 0.5% of all data sent by the user, be transmitted to the final destination using the same supposed compromised TOR EXIT NODE, during the 34 minutes and 14 seconds of the tests. The presented results show that the SPLITTER approach will considerably difficult the correlation and the traffic analyses inside TOR network because the adversary will need more time or effort to:

1. Collect the minimum necessary amount of data for each compromised TOR RELAY/NODE before be able to correlate the data. More statistical analyses could be necessary to reach the same result.

2. Compromised more TOR RELAYS/NODES in different parts of the globe, to keep a sustainable average about the total amount of data collected around the world.

3. Find the injected pattern which hopefully has been spread among all countries used during the transmission process, and be able to correlate the time, considering the deliberated disturb created by SPLITTER.

4. Correlate the pieces of intercepted message, considering the adversary ability to compromise the TOR encryption layers. The pieces of message collected in the different parts of the globe will be in a random order and now the adversary has a huge puzzle to mount.

5. Find the real geolocation of the victim considering that the victim is not connecting directly to TOR but using a combination of VPS and VPN.

Based on the great stability and performance provided by HAPROXY inside the SPLITTER context. The TOR PROJECT can consider a similar solution to enhance the performance of TOR BROWSER and promote a better load balancing between the TOR RELAYS.

The results presented here are questionable considering all limitations, however, every single de-anonymization technique mentioned in this paper should be tested against the results provided by the SPLITTER approach. The overall success to identify the source of the TOR requests in the real-world experiments provided by these techniques now becomes questionable:

- Are the adversaries capable to keep the same precision after a global spread of the victim requests?

- The delay between the requests sent using the same compromised NODE, still allowing the adversaries to correlate and find the injected pattern?

- How many compromised TOR relays each de-anonymization technique will demand to keep the same precision?

Many improvements could be done in the rustic solution that was proposed in this paper, however, the average of 0.5% demonstrated here could set a new bar for the future de-anonymization techniques.

REFERENCES

[1] Sambuddho Chakravarty, Marco V. Barbera, Georgios Portokalidis, Michalis Polychronakis and Angelos D. Keromytis - "On the Effectiveness of Traffic Analysis Against Anonymity Networks Using Flow Records".
[Online] Available: https://mice.cs.columbia.edu/getTechreport.php?techreportID=1545&format=pdf

[2] Nathan S. Evans, Roger Dingledine and Christian Grothoff - "A Practical Congestion Attack on Tor Using Long Paths".
[Online] Available: https://www.usenix.org/legacy/event/sec09/tech/full_papers/evans.pdf

[3] Matthew Wright, Micah Adler, Brian N. Levine and Clay Shields - "An analysis of the degradation of anonymous protocols"
[Online]. Available: http://people.cs.georgetown.edu/~clay/research/pubs/wright.ndss01.pdf

[4] Nicholas Hopper, Eeugene Y. Vasserman, and Eric Chan-Tin - "How Much Anonymity does Network Latency Leak?".
[Online] Available: https://www-users.cs.umn.edu/~hoppernj/tissec-latency-leak.pdf

[5] Sebastian Zander and Steven J. Murdoch - "An Improved Clock-skew Measurement Technique for Revealing Hidden Services".
[Online] Available: https://www.usenix.org/legacy/event/sec08/tech/full_papers/zander/zander.pdf

[6] Kevin Bauer, Damon McCoy, Dirk Grunwald, Tadayoshi Kohno and Douglas Sicker -
"Low-Resource Routing Attacks Against Anonymous Systems"
[Online] Available: http://www.cs.colorado.edu/department/publications/reports/docs/CU-CS-1025-07.pdf

[7] TOR project official web site. [Online] Available: https://www.torproject.org/

[8] TOR project overview. [Online] Available: https://www.torproject.org/about/overview.html.en

[9] "Statistical Analysis Handbook".
[Online] Available: http://www.statsref.com/StatsRefSample.pdf

[10] Steven J. Murdoch and George Danezis - "Low-Cost Traffic Analysis of Tor".
[Online] Available: https://murdoch.is/papers/oakland05torta.pdf

[11] FBI Official web site - "Dozens of Online 'Dark Markets' Seized Pursuant to Forfeiture Complaint Filed in Manhattan Federal Court in Conjunction with the Arrest of the Operator of Silk Road 2.0".
[Online] Available: https://www.fbi.gov/contact-us/field-offices/newyork/news/press-releases/dozens-of-online-dark-markets-seized-pursuant-to-forfeiture-complaint-filed-in-manhattan-federal-court-in-conjunction-with-the-arrest-of-the-operator-of-silk-road-2.0

[12] FBI Official web site - "Operator of Silk Road 2.0 Website Charged in Manhattan Federal Court" [Online]. Available: https://www.fbi.gov/contact-us/field-offices/newyork/news/press-releases/operator-of-silk-road-2.0-website-charged-in-manhattan-federal-court

[13] FBI Special Agent: Thomas M. Dalton report about the hoax bomb in Harvard University resulting in the prison of Eldo Kim.
[Online] Available: https://cbsboston.files.wordpress.com/2013/12/kimeldoharvard.pdf

[13.1] FBI Official web site - "Harvard Student Charged with Bomb Hoax".
[Online] Available: https://archives.fbi.gov/archives/boston/press-releases/2013/harvard-student-charged-with-bomb-hoax

[14] FBI Official web site - "Six Hackers in the United States and Abroad Charged for Crimes Affecting Over One Million Victims". [Online] Available: https://archives.fbi.gov/archives/newyork/press-releases/2012/six-hackers-in-the-united-states-and-abroad-charged-for-crimes-affecting-over-one-million-victims

[15] Adrian Crenshaw - “Dropping Docs on Darknets: How People Got Caught - Defcon 22”
[Online] Available: https://www.youtube.com/watch?v=7G1LjQSYM5Q

[16] TOR Official Project web site - Metrics about TOR network.
[Online] Available: https://metrics.torproject.org/networksize.html

[17] TOR Official Project web site - “Tor: Onion Service Protocol”.
[Online] Available: https://www.torproject.org/docs/onion-services.html.en

[18] Steven J. Murdoch - “Hot or Not: Revealing Hidden Services by their Clock Skew”.
[Online] Available: https://murdoch.is/papers/ccs06hotornot.pdf

[19] TOR Official Project web site - “Who Uses Tor?”.
[Online] Available: https://www.torproject.org/about/torusers.html.en

[20] Rob Jansen, Marc Juarez, Rafa Gálvez, Tariq Elahi and Claudia Diaz - "Inside Job: Applying Traffic Analysis to Measure Tor from Within".
[Online] Available: https://www.robgjansen.com/publications/insidejob-ndss2018.pdf

[21] TOR project official web site - FAQ: "What are Entry Guards?".
[Online] Available: https://www.torproject.org/docs/faq#EntryGuards

[22] TOR project offical blog - "Improving Tor's anonymity by changing guard parameters".
[Online] Available: https://blog.torproject.org/improving-tors-anonymity-changing-guard-parameters

[23] Free Haven – Online Anonymity Papers Library.
[Online] Available: https://www.freehaven.net/anonbib/

[24] TOR project offical blog - "Research problem: better guard rotation parameters"
[Online] Available: https://blog.torproject.org/research-problem-better-guard-rotation-parameters

[25] Nick Mathewson - "Cryptographic Challenges in and around Tor".
[Online] Available: https://crypto.stanford.edu/RealWorldCrypto/slides/tor.pdf

[26] TOR project official web site – FAQ: “How often does Tor change its paths?”
[Online] Available: https://www.torproject.org/docs/faq#ChangePaths

[27] TOR project official web site – TOR MANUAL
[Online] Available: https://www.torproject.org/docs/tor-manual.html.en

[28] Milad Nasr, Amir Houmansadr and Arya Mazumdar - "Compressive Traffic Analysis:A New Paradigm for Scalable Traffic Analysis".
[Online] Available: https://people.cs.umass.edu/~milad/papers/compress_CCS.pdf

[29] ISACA - “Geolocation: Risk, Issues and Strategies”.
[Online] Available: https://www.isaca.org/Groups/Professional-English/wireless/GroupDocuments/Geolocation_WP.pdf

[30] Eugene Gorelik - “Cloud Computing Models”
[Online] Available: https://web.mit.edu/smadnick/www/wp/2013-01.pdf

[31] Alexa Huth and James Cebula - “The Basics of Cloud Computing”
[Online] Available: https://www.us-cert.gov/sites/default/files/publications/CloudComputingHuthCebula.pdf

[32] Jason A. Donenfeld - “WireGuard: Next Generation Kernel Network Tunnel”
[Online] Available: https://www.wireguard.com/papers/wireguard.pdf

[33] HAPROXY – Official Web Site.
[Online] Available: https://www.haproxy.org/

[34] Privoxy – Official Web Site
[Online] Available: http://www.privoxy.org/

[35] TOR Standalone Linux version Download Page.
[Online] Available: https://www.torproject.org/download/download-unix.html.en

[36] HAPROXY - Documentation.
[Online] Available: https://www.haproxy.org/#docs

[37] PRIVOXY - Official User Manual.
[Online] Available: http://www.privoxy.org/user-manual/index.html

[38] King, Kevin, "Personal Jurisdiction, Internet Commerce, and Privacy: The Pervasive Legal Consequences of Geolocation Technologies," Albany Law Journal of Science and Technology, January 2011

[39] Viviane Reding - "Digital Sovereignty: Europe at a Crossroads"
[Online] Available: http://institute.eib.org/wp-content/uploads/2016/01/Digital-Sovereignty-Europe-at-a-Crossroads.pdf

[40] Tim Maurer, Robert Morgus, Isabel Skierka, Mirko Hohmann - "Technological Sovereignty: Missing the Point?"
[Online] Available: http://www.digitaldebates.org/fileadmin/media/cyber/Maurer-et-al_2014_Tech-Sovereignty-Europe.pdf

[41] BSD License Definition
[Online] Available: http://www.linfo.org/bsdlicense.html

[42] Joel Reardon and Ian Goldberg - "Improving Tor using a TCP-over-DTLS Tunnel"
[Online] Available: https://www.usenix.org/legacy/event/sec09/tech/full_papers/reardon.pdf

[43] TOR Metrics – Official WebSite.
[Online] Available: https://metrics.torproject.org/rs.html#search/country:ES%20flag:exit

[44] Docker – Official Documentation
[Online] Available: https://docs.docker.com/

[45] Docker – Official Documentation "Expose (incoming ports)"
[Online] Available: https://docs.docker.com/engine/reference/run/#expose-incoming-ports