SnoopyBot: An Android spyware to bridge the mixes in Tor

Evangelos Mitakidis, Dimitrios Taketzis, Alexandros Fakis, Georgios Kambourakis
Department of Information and Communication Systems Engineering,
University of the Aegean, Karlovassi, Greece

***Abstract*—We present a moderately simple to implement but very effective and silent deanonymization scheme for Tor traffic. This is done by bridging the mixes in Tor, that is, we control both the traffic leaving the Onion Proxy (OP) and the traffic entering the Exit node. Specifically, from a user's viewpoint, our proposal has been implemented in the popular Android platform as a spyware, having the dual aim to manipulate user traffic before it enters the Tor overlay and explicitly instruct OP to choose an exit node that is controlled by the attacker. When the user traffic is received by the rogue exit node it is filtered, and the sender's IP details become visible. Notably, apart from deobfuscating normal http traffic, say, send via the Tor browser, the proposed scheme is able to manipulate https requests as well.**

I. Introduction

Internet was not created and designed with anonymity in mind [1]. So, normally, if two parties want to communicate it is mandatory to provide their routable source IP addresses to each other. This situation however is subject to traffic analysis by potential eavesdroppers. Typically, disclosure of IP address is necessary across all applications communicating over IP protocol, however most of the well-established network security protocols fail to provide full anonymity. For instance, Transport Layer Security (TLS) succeeds in perfectly encrypting and authenticating packet contents, but fails to protect the identities of the involved parties. On the other hand, IPsec achieves to protect the anonymity of the communicating parties, but this stands true only when used in ESP/Tunnel mode and only for data traveling between the two ends of the tunnel. That is, the links between the sender and the one end of the tunnel, and between the other end of the tunnel and the final destination do not afford protection from traffic analysis. However, to fully block traffic analysis, one needs a solution which will provide anonymity in a cross-layer fashion.

One of the most prominent anonymization systems today is the Onion Router network (TOR) [2], which consists of a group of volunteer-operated servers, each of them working as a proxy. Its main purpose is to allow individuals to protect their anonymity, by means of deterring traffic analysis. In fact, Tor is an overlay network that among others can be used as a censorship circumvention tool. Namely, Tor is particularly useful to certain parties, including journalists trying to communicate safely with whistle-blowers or dissidents, activist groups that endorse maintaining civil liberties online, and people living in countries with restricted freedom of information and access to websites [3]. On the other hand, Tor can also be used for malicious purposes. Apart from using Tor with the aim of obfuscating the forensic signal behind attacks like Denial of Service (DoS), aggressors employ Tor's capability to build the so-called hidden services to accomplish illegal or malicious actions, such as drug dealing or illegal gun trafficking. This has spurred research on methods to deanonymize Tor traffic. Usually, such deanonymization schemes focus on the exploitation of the Exit node. Nevertheless, better results can be achieved if traffic analysis performed at the exit node is bridged with either the Entry node or an even precedent point of entry, namely before the outgoing data reach the Onion Proxy (OP).

Our Contribution: In this paper, we present a new stealthy way of instantly deanonymizing specific or random Tor users. Specifically, the proposed attack relies on (a) a user-side spyware called SnoopyBot we specifically created for the Android platform, and (b) on the very plausible assumption that the attacker is in control of at least one Tor Exit node. Precisely, among others, SnoopyBot's main purpose is to inject user-identifying information in every request the infected user channels through the Tor network. In this way, when the rogue Exit node receives the corresponding request, the sender's true identity in terms of their public IP address will be revealed. Putting it another way, by controlling both ends of the circuit, SnoopyBot achieves to bridge the mixes in Tor. This also means that the assault does not capitalize on an inherent Tor vulnerability, but entirely on the ability of the attacker to control the two ends of the Tor circuit.

The rest of this paper is structured as follows. The next section offers a succinct discussion on Tor. Section III details on our implementation. The limitations of our scheme as well as possible countermeasures are given in Section IV. Section V briefly presents the related work. The last section concludes the paper and presents directions for future work.

II. Tor Operation

Today, Tor is probably the most popular open source implementation of a distributed overlay network which aims to anonymize TCP-based traffic. The Tor software installed on the client device, namely Onion Proxy (OP), chooses a random path through the available Onion Routers (OR) in the network and constructs a circuit, in which each OR in the path knows only its predecessor and successor, but no other nodes. After that, the user's traffic flows down the selected circuit using onion routing which is based on layered encryption. To further impede the analysis of data streams based on

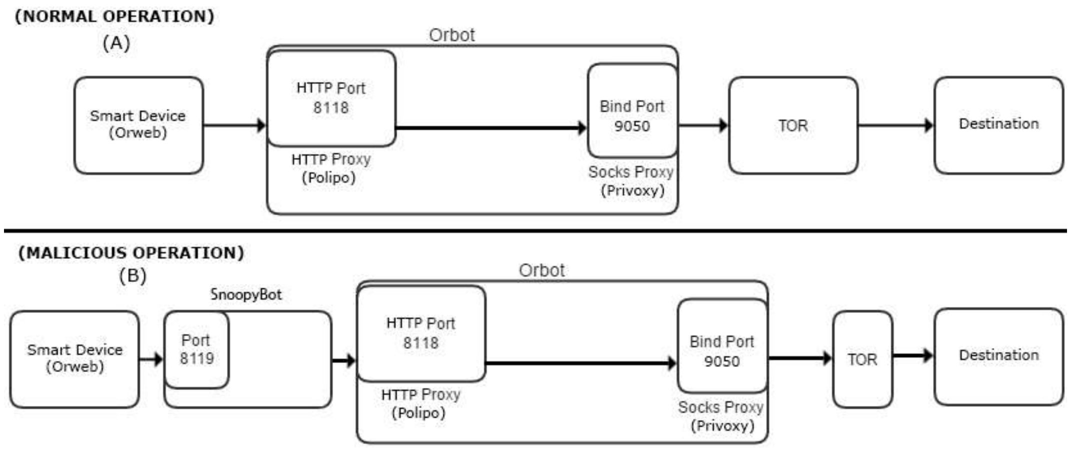

Fig. 1: Tor operation at the client side: (a) Normal, (b) After infection by SnoopyBot

traffic characteristics, Tor overlay uses cells of fixed size to communicate its users' data among the different ORs. Overall, this approach blocks traffic analysis because no single entity is aware of the full path a packet has traveled.

A Tor user, say, Alice needs to have locally installed the Tor software, which in a desktop environment is the Tor browser. In Android environment, Tor software is provided by means of two separate apps, namely, Orbot [4] and Orweb [5]. Any mobile browser that offers http proxy support can also be used with Orbot if properly configured, but Orweb is configured out-of-the-box. Orbot is used to create an OP and construct a circuit to realize the anonymization of client data into the Tor overlay. The necessary information for the available ORs is obtained from the Tor's Directory Server. The ORs are proxies that relay data between the two endpoints, Alice and the actual destination, say, a web server. Each OR maintains a TLS connection to other ORs in the constructed circuit. By default a Tor circuit consists of three ORs known as Entry/Guard Node, Relay Node, and Exit Node. Typically, to construct a circuit, Alice's OP chooses three ORs and starts negotiating a symmetric key with each one of them by means of Diffie-Hellman handshakes. After the circuit is ready, a service, e.g., web browsing, is achieved when Alice's browser establishes a TCP connection with the corresponding web server. To avoid DNS resolution queries (which may reveal information about Alice's identity) sent directly by the web browser over the Internet, Alice's OP uses an HTTP proxy so that her traffic can be diverted through Tor. Therefore, in a properly configured Tor installation, the exit node is responsible to make DNS resolution of, say, the user's http requests.

III. Implementation

A. Architecture

The architecture of our implementation consists of two main components. The first one, is a user's side spyware, coined as SnoopyBot, which is essentially a malicious Android application that must be installed on user's smart device. At present, SnoopyBot is developed for Android v4.4 and has also been successfully tested on Android 5.1.1.

The second component of our architecture is at least one specially configured Tor Exit node that sniffs user traffic passing through it. When the Exit node receives data having the signature of SnoopyBot it instantly logs and deanonymizes it.

It is to be emphasized that for the needs of the work at hand the guidelines for ethical Tor research [6] were strictly followed. Specifically, SnoopyBot was developed and used on the test smartphone and the only traffic that was logged on the Exit node was solely the one stemming from that smartphone.

B. SnoopyBot

At a high level, assuming an http or https connection, SnoopyBot has three main goals. First, it modifies the default settings of any Tor application running on the smart device and are necessary to access Tor network. As already pointed out, these applications are, Orbot and Orweb. Second, it obtains the public IP of the user, and third, it hijacks (acting as a man-in-the-middle) the connection in order to inject user's personal information to the requested URL.

SnoopyBot is designed to instantly launch right after its installation, as well as after every reboot of the smart device. Upon its successful installation, SnoopyBot masquerades itself as Adobe Flash v2.0, which is a well-known software and will normally present itself as a benign application to the owner of the smart device. Also, it removes any application (SuperSU, Superuser) that gives root permissions to the user, including pop-up permission request dialogs. Pop-up dialogs are shown to the user with the use of toast messages, every time an application requests root permissions. This kind of action can attract the victim's attention, and such a suspicion could eventually lead to have SnoopyBot uninstalled. In addition, we particularly concentrated on the stealthy operation of SnoopyBot to make harder its detection by anti-spyware software. Towards this goal, we minimized the number of connections SnoopyBot makes outside the Tor network. More specifically, SnoopyBot generates only one request to a public web service, to obtain the user's public IP, while the rest of the communication remains within the Tor network. Naturally,

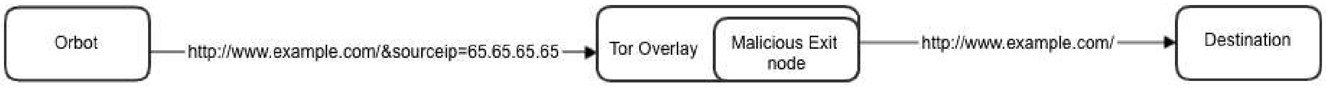

Fig. 2: High-level description of the attack for a http request

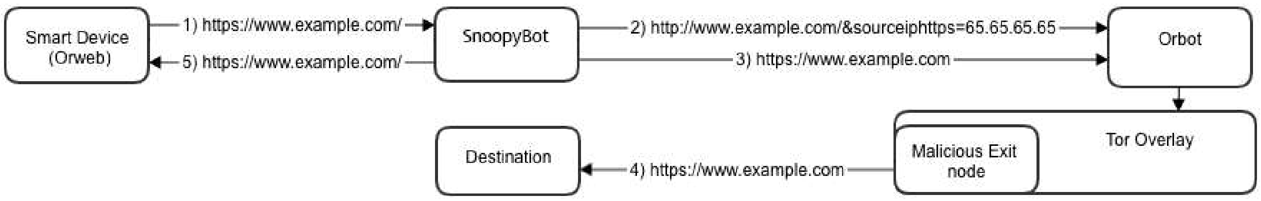
Smart Device (Orweb)
1) https://www.example.com/
5) https://www.example.com/
SnoopyBot
2) http://www.example.com/&sourceiphttps=65.65.65.65
3) https://www.example.com
Orbot
Tor Overlay
Malicious Exit node
4) https://www.example.com
Destination

Fig. 3: High-level description of the attack for a https request (basic steps)

every time the IP of the user changes the same request must be repeated.

As already pointed out, the spyware needs to modify Tor settings at the client side. Figure 1A depicts a typical http or https GET transaction when processed by Tor at the client side. The spyware invades this procedure and modifies Orbot configuration settings inside the torrc file, to always use the attacker's exit node during the creation of any Tor circuit. After that, it prohibits any further modifications to torrc. It is to be noted that Orbot consists of two main components. The Polipo [7] http proxy that listens on port 8118, and Privoxy SOCKS proxy [8] that listens on port 9050. Essentially, Polipo forwards the http traffic to Privoxy, which subsequently forwards the traffic to Tor. Additionally, as depicted in fig. 1B, SnoopyBot changes the http proxy settings of Orweb to point to destination port 8119 instead of the default 8118. This forces all victim traffic to be proxied through the SnoopyBot HTTP proxy. After manipulating the GET request, SnoopyBot will forward it to HTTP port 8118 as normal, which is Orbot's HTTP Port. Putting it another way, SnoopyBot HTTP proxy acts as a man-in-the-middle between Orweb and Orbot, thus it is able to sniff and modify any data passing through this link. After that, SnoopyBot triggers a connection to a public service (e.g., www.myip.com) to get the public IP of the user (victim). Since this is the only outbound connection made by the spyware, its footprint on the system is minimal. Having the public IP of the victim, the SnoopyBot proxy injects it along with a certain identification string (i.e., SnoopyBot's signature) into all URLs requested by the victim.

For example, assuming that the requested URL is http://www.example.com before injection, it will be modified to http://www.example.com/&sourceip=65.65.65.65 and be forwarded to Orbot. Note that SnoopyBot signature's in this case is the "&sourceip" string. Of course, this string is entirely up to the spyware coder and can be changed between the different versions of the spyware. As illustrated in fig. 2, this signature will be removed at the Exit node. Also, as shown in fig. 3, in case a HTTPS request is detected, SnoopyBot momentarily blocks it and then constructs and forwards an identical HTTP one, injected with the victim's public IP as normal. At nearly the same time, it forwards the original HTTPS request to Orbot to be dispatched via the Tor overlay. This time however, SnoopyBot's signature in the http bogus request is changed to "&sourceiphttps". As explained further down in section III.C, this alteration will enable the Exit node to log and then discard the corresponding GET message. After installing SnoopyBot on a smartphone, we have performed several scans using popular mobile anti-virus applications (including Kasperksy, Bitdefender and CleanMaster) and all of them showed no sign of infection.

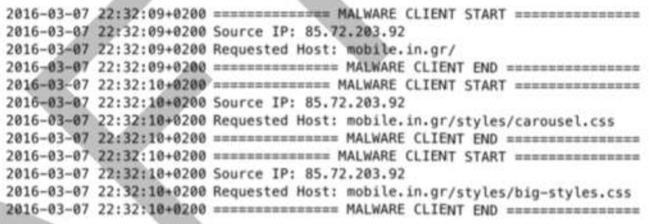
```
2016-03-07 22:32:09+0200 =============== MALWARE CLIENT START ===============
2016-03-07 22:32:09+0200 Source IP: 85.72.203.92
2016-03-07 22:32:09+0200 Requested Host: mobile.in.gr/
2016-03-07 22:32:09+0200 ================ MALWARE CLIENT END ================
2016-03-07 22:32:10+0200 =============== MALWARE CLIENT START ===============
2016-03-07 22:32:10+0200 Source IP: 85.72.203.92
2016-03-07 22:32:10+0200 Requested Host: mobile.in.gr/styles/carousel.css
2016-03-07 22:32:10+0200 ================ MALWARE CLIENT END ================
2016-03-07 22:32:10+0200 =============== MALWARE CLIENT START ===============
2016-03-07 22:32:10+0200 Source IP: 85.72.203.92
2016-03-07 22:32:10+0200 Requested Host: mobile.in.gr/styles/big-styles.css
2016-03-07 22:32:10+0200 ================ MALWARE CLIENT END ================
```

Fig. 4: Filtered log file records at the Exit node for http traffic

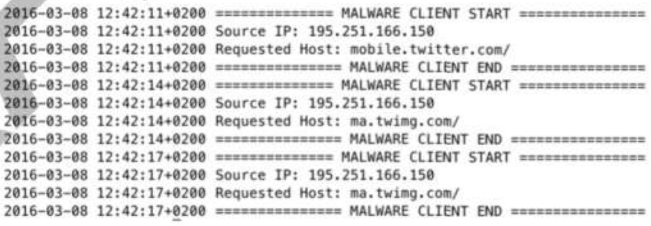
```
2016-03-08 12:42:11+0200 =============== MALWARE CLIENT START ===============
2016-03-08 12:42:11+0200 Source IP: 195.251.166.150
2016-03-08 12:42:11+0200 Requested Host: mobile.twitter.com/
2016-03-08 12:42:11+0200 ================ MALWARE CLIENT END ================
2016-03-08 12:42:14+0200 =============== MALWARE CLIENT START ===============
2016-03-08 12:42:14+0200 Source IP: 195.251.166.150
2016-03-08 12:42:14+0200 Requested Host: ma.twimg.com/
2016-03-08 12:42:14+0200 ================ MALWARE CLIENT END ================
2016-03-08 12:42:17+0200 =============== MALWARE CLIENT START ===============
2016-03-08 12:42:17+0200 Source IP: 195.251.166.150
2016-03-08 12:42:17+0200 Requested Host: ma.twimg.com/
2016-03-08 12:42:17+0200 ================ MALWARE CLIENT END ================
```

Fig. 5: Filtered log file records at the Exit node for https traffic

C. *Exit node*

For the exit node, we used the official Tor software on a Linux Ubuntu server, and configured the torrc file so as for our server to be able to operate as a trusted Exit node. In particular, we configured the torrc file in the Exit node to route HTTP and HTTPS traffic (ExitPolicy accept *:80 and ExitPolicy accept *:443). We also adjusted the amount of bandwidth that will be made available to Tor and finally provided a name for the Exit node.

Moreover, a Python HTTP proxy was implemented. This proxy listens on a different port on the same box as Tor and eavesdrops on incoming traffic. By using iptables we redirected all the incoming traffic with destination port 80 or 8080 (http-alt) to our HTTP proxy. As observed from fig. 3, upon the reception of any http request that carries one of the SnoopyBot's signatures, the HTTP proxy logs it, strips the signature from the request, and forwards the request to its destination, say, a webserver as normal. An exception to this rule is any incoming bogus http request which is blocked in order for the destination not to receive an extra http request corresponding to the original https one. As explained in section III.B, all bogus http requests carry a special SnoopyBot's signature. The HTTP response from the webserver is forwarded to the Tor network with no further manipulation.

Finally, as shown in fig. 4 and 5, a bash script was created to filter the logged traffic and present the results containing the victim's source IP to the attacker.

IV. Limitations & Countermeasures

We acknowledge the following three limitations that the current version of SnoopyBot faces. Actually, the first two are considered out of scope of the paper at hand and are left for future work.

- SnoopyBot needs to somehow infect and spread amongst users. One way to do so is to bundle the spyware apk setup file with another apk, belonging to an app that requires root permissions during installation. The latter apk may belong to a legitimate app that the user would download from an alternative Android app market. Using this method, during apk installation, the root permissions that the legitimate app would ask from the user for performing its tasks would also be given to the spyware after installation. A second way of spreading SnoopyBot is to make an on-the-fly bundle and injection of the spyware when a user makes a request to our Exit node for an .apk download. This method however, increases the chances of the malevolent Exit node to be detected by Tor. It is also implied that the infection of specific users is considered more difficult than spreading SnoopyBot among the public at large.
- Currently, SnoopyBot works only with Orweb. However, there are several other web browsers that can be used in cooperation with Orbot, and thus SnoopyBot needs to be modified to co-work with each one of them.
- The user's smart device must be rooted. This condition is necessary for the spyware to perform its actions.

Basically, to cope with the attack described in this paper, one needs to take precautions for both the Exit node and the mobile device. This is because the only thing the attack does is to build a covert channel between the mobile device and the Exit node, which are both controlled by the attacker. Regarding the Exit node, traffic sniffing is very hard to detect; practically, there is no safe way for the Tor network to tell if an Exit node monitors traffic (and of course this is not a Tor insecurity). This means that the only effective countermeasure against a rogue Exit node is to only use Exit nodes that are known to be trusted, at least to certain degree (e.g., by consulting [9]).

As already pointed out, SnoopyBot requires a rooted device, which in turn means that a non-rooted device is not vulnerable. If the mobile device is rooted, then the settings for Orbot can be stored in an encrypted database to avoid being manipulated by an evil-doer. Barring the situation that a great number of malicious Exit nodes exist, if SnoopyBot cannot access Orbot settings, the probability for a mobile user to get infected by SnoopyBot and use a malicious Exit node is tiny. Naturally, another defensive measure is to alert the end-user upon detecting any unprovoked change to the mobile browser settings. This would be enough to raise the user's suspicion, making them to initiate a device scan for malicious apps. Also, the user should avoid installing any app that is not in, say, the Play Store and/or its origin is nebulous.

Lastly, provided that there is an Intrusion Detection System (IDS) installed in the device (or on the same network with the mobile device), there should be an alert for any internal IP that queries sites for obtaining the user's public IP, especially if it is a frequent request from the same internal IP.

V. Related Work

According to the literature, the most powerful deanonymization attacks on Tor are based on some scheme that makes possible the correlation of the data between two points (preferably the Entry and Exit node) of the Tor overlay. This strategy is also followed by SnoopyBot. In the following, we briefly discuss the related work and point out the additional contributions introduced by SnoopyBot.

It is true that the weakest node in a Tor circuit is the Exit node. This is because at this node the actual data send by the Tor user are decrypted and forwarded to its final destination. An Exit node can behave in a malicious way by various means and plays an essential role to the identification of a Tor user. For instance, as described by FortConsult Security [10], it can be used for injecting software that will be executed by the victim's browser. Another approach that makes use of a malicious Exit node was given by X. Wang et al. [11]. In their approach, the authors injected a forged web page back to the Entry node (forged web page injection attack) or they modified the HTTP traffic passing through it, thus exercising a web page modification attack. Other contributions like the one described by Z. Ling et al. [12] concentrate on timing attacks exercised by a malicious Entry node. Similarly to the previous one, the scheme proposed by X. Fu et al. [13] places the attacker at both a malicious entry and exit OR. The aggressor manipulates the outbound cells stemming from the various OPs, causing them to be determined at the exit node, and thus eventually confirming the communication relationship between the sender and the receiver.

A second category of deanonymization methods capitalizes on web browsers with enabled Javascript, Flash or other similar technologies. Although these features are designed to enhance user's experience when browsing, they are potentially harmful to privacy, as described by T. Abbott et al. [14]. An attack taking advantage of this fact can use a malicious Exit node for modifying web pages through the injection of JavaScript code that repeatedly connects to a logger server sending a distinctive signal.

To sum up, most of the attacks on the bibliography are mainly based on infected Entry or Exit nodes, considering that even security-savvy users are not able to detect any untrustworthy node in the Tor Network. Whereas the cornerstone of SnoopyBot's operation is also the interlinking of two Tor endpoints, the way it does so differs from the related work in two major aspects. First, the manipulation of data is performed at the client before the user data reach OP. Secondly, it does not require the attacker to possess special equipment or a large mass of computing resources. On top of that, its operation at both ends, i.e., the client and the Exit node, is as silent as possible.

VI. Conclusion

Tor's major weakness is the Exit node of each circuit, as all traffic that passes through this node is potentially unprotected. This shortcoming attracted several researchers to develop methods of tracking and avoiding malicious Exit nodes. However, as with SnoopyBot, this is hard to achieve in cases where the Exit node just silently logs the traffic passing via it, leaving no other trace of its privacy-invasive activity. In this case, countermeasures need to be taken on the client side as well. For instance, the apps that provide access to Tor, like Orbot or Orweb, must encrypt their settings or use a secure database for storing them. In any case, however, the root cause of the Exit node's problem is not due to the internal workings of Tor, but to end-users not employing https connections or other means of protection at the application layer. This problem is even aggravated by badly configured web browsers or other applications and the rise of privacy-invasive software as in our case.

Moreover, one of the most prominent security issues that can occur during Tor installation on, say, a smartphone is the root permissions Tor requires to anonymize outgoing traffic stemming from any application other than Tor's official browser. This requirement leads many users to root their smart device, which, as a direct consequence gives the ability to any malicious application to gain access to critical files on the Android system.

In this paper, we presented a spyware targeting the Android platform. SnoopyBot is able to infect Tor users with the purpose of deanonymizing them. After this spyware gets installed on the device, it has the ability to change the settings of torrc file according to attacker's will. After deactivating any root application that can notify user of its actions, SnoopyBot alters the Exit node option, setting one or more Exit nodes that are controlled by the attacker. Having successfully changed the torrc settings, the spyware starts working as a proxy. In every request the user makes, the proxy intervenes, and adds an extra field containing the victim's public IP and one of the spyware signatures in it. The request is forwarded to Tor network and passes through the rogue Exit node, which logs it. In that way, every user infected by SnoopyBot can be deanonymized. Once more, it is stressed that SnoopyBot has been developed with stealth operation in mind. That is, while a malware in general may have numerous capabilities (including user deanonymization), most of them need to connect to some outside server to fully perform their operations, which renders them more easily detectable by defensive mechanisms, such as IDS or anti-malware software. SnoopyBot, on the other hand, does not create extra traffic as it avoids any communication with C&C servers (both inside and outside Tor network). Instead, it uses the malicious Exit node per se as a C&C server, which however is a valid OR and is explicitly selected for infected clients. Therefore, from a defensive mechanism point of view, the traffic seems perfectly legitimate.

Our intention is to extend this work by creating more Exit nodes and updating the user-side spyware, so it will be able to alter periodically the address of the Exit node, e.g., each time a new Tor circuit is created. This would further obfuscate SnoopyBot's trails and make its detection harder. An additional direction for future work would be the implementation of a covert communication channel between the Exit node and SnoopyBot. Such a covert channel can capitalize on the parameters carried by http requests and replies for achieving private extensive information leakage. Last but not least, it would be interesting for one to modify SnoopyBot operation to scan for and modify accordingly all browsers found on the device after every boot. In this way, irrespective of the web browser the user employs, SnoopyBot will be in position to exercise the attack.

References

[1] G. Kambourakis, "Anonymity and closely related terms in the cyberspace: An analysis by example," *Journal of information security and applications*, vol. 19, no. 1, pp. 2–17, 2014.

[2] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 21–21. [Online]. Available at: http://dl.acm.org/citation.cfm?id=1251375.1251396

[3] G. Karopoulos, A. Fakis, and G. Kambourakis, "Complete sip message obfuscation: Privasip over tor," in *Availability, Reliability and Security (ARES), 2014 Ninth International Conference on*. IEEE, 2014, pp. 217–226.

[4] Guardian-Project. Orbot: Tor for android. [Online]. Available at: https://guardianproject.info/apps/orbot/

[5] Guardian-Project. Orweb: Private web browser. [Online]. Available at: https://guardianproject.info/apps/orweb/

[6] I. The-Tor-Project. Ethical tor research: Guidelines. [Online]. Available at: https://blog.torproject.org/blog/ethical-tor-research-guidelines

[7] J. Chroboczek. Polipo - a caching web proxy. [Online]. Available at: https://www.irif.univ-paris-diderot.fr/ jch/software/polipo/

[8] Privoxy-Developers. Privoxy. [Online]. Available at: https://www.privoxy.org

[9] J. B. Kowalski. Torstatus - tor network status. [Online]. Available at: https://torstatus.blutmagie.de/

[10] A. Christensen, "Practical onion hacking: finding the real address of tor clients," *FortConsults advisory http://www.fortconsult.net/images/pdf/Practical_Onion_Hacking.pdf*, 2009.

[11] X. Wang, J. Luo, M. Yang, and Z. Ling, "A potential http-based application-level attack against tor," *Future Generation Computer Systems*, vol. 27, no. 1, pp. 67–77, 2011.

[12] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A new cell counter based attack against tor," in *Proceedings of the 16th ACM conference on Computer and communications security*. ACM, 2009, pp. 578–589.

[13] X. Fu, Z. Ling, J. Luo, W. Yu, W. Jia, and W. Zhao, "One cell is enough to break tors anonymity," in *Proceedings of Black Hat Technical Security Conference*, 2009, pp. 578–589.

[14] T. G. Abbott, K. J. Lai, M. R. Lieberman, and E. C. Price, "Browser-based attacks on tor," in *Privacy Enhancing Technologies*. Springer, 2007, pp. 184–199.