TOR-BASED MALWARE AND TOR CONNECTION DETECTION

Ibrahim Ghafir *, Jakub Svoboda † and Vaclav Prenosil *

* Faculty of Informatics, Masaryk University
† Institute of Comouter Science, Masaryk University
Brno, Czech Republic
ibrahim_ghafir@hotmail.com, svoboda@ics.muni.cz, prenosil@fi.muni.cz

Keywords: Cyber attacks, bonet, Tor network, malware, intrusion detection system.

Abstract

Anonymous communication networks, like Tor, partially protect the confidentiality of user traffic by encrypting all communications within the overlay network. However, Tor is not only used for good; a great deal of the traffic in the Tor networks is in fact port scans, hacking attempts, exfiltration of stolen data and other forms of online criminality. The anonymity network Tor is often misused by hackers and criminals in order to remotely control hacked computers. In this paper we present our methodology for detecting any connection to or from Tor network. Our detection method is based on a list of Tor servers. We process the network traffic and match the source and destination IP addresses for each connection with Tor servers list. The list of Tor servers is automatically updated each day and the detection is in the real time. We applied our methodology on campus live traffic and showed that it can automatically detect Tor connections in the real time. We also applied our methodology on packet capture (pcap) files which contain real and long-lived malware traffic and we proved that our methodology can successfully detect Tor connections.

1 Introduction

Large scale Internet censorship by state-level authorities [1] has spurred the development of new privacy enhancing technologies that circumvent the censor, followed by new techniques to recognize and block these circumvention tools [2]. Tor [3] is the most popular deployed system for fighting censorship and online privacy encroachments, currently supporting several hundreds of thousands of users daily and transferring roughly 3 GiB/s in aggregate [4]. Tor uses onion routing [5] to route *clients'* traffic through a *circuit* of geographically diverse *relays,* preventing any single relay from linking the client to its Internet destination.

Tor is a distributed overlay network designed to anonymize TCP-based applications like web browsing, secure shell and instant messaging [3], or more simply, an anonymity network. Tor uses the concept of 'onion routing' [6], which itself is based on Chaum's original Mix-Net design [7].

To browse the World Wide Web (WWW) anonymously, a client first requests a list of available Tor routers from one of the directory servers [8]. Once received, the client then initiates a path by sending a message to the first Tor router (Entry Guard), and using Diffie-Hellman key exchange [9], a session key is generated between the client and Entry Guard. The circuit is then extended by undertaking the same process one router (hop) at a time, incrementally extending each time with the established session keys for the previous hop(s), a technique also known as 'telescoping'. Once three hops have been established, the circuit is then complete and ready to be used for Internet traffic.

The 'core' messages from the Tor client, e.g. HTTP GET requests, are encapsulated alongside an Internet Protocol (IP) header for the next hop, within individually encrypted layers, creating the multi-layered Tor 'onion'. The onion is then forwarded through the SOCKS proxy interface (local host), and relayed by one of the available Tor circuits, as data-streams, via multiplexed TCP connections between the Tor routers. The onion is then 'un-peeled' incrementally, by the Tor router at each hop, revealing the next layer until the core message is exposed at the final hop (i.e. Exit Router), where it is sent to, and processed at the final destination e.g. web server.

The overall of design of Tor, including the use of three hops and ephemeral (short-lived) session keys, helps maintain anonymity through, the concept known as, 'perfect forward secrecy' [3].

While preventing someone who is monitoring your Internet connection from learning what sites you visit has positive connotations for free speech in many parts of the world, it presents a problem for corporate and educational environments where identification and control of sites accessed is necessary to enforce acceptable use policies, minimize data leakage, and stop users from accessing harmful content.

Tor is, however, not only used for good; a great deal of the traffic in the Tor networks is in fact port scans, hacking attempts, exfiltration of stolen data and other forms of online criminality [10]. The anonymity network Tor is often misused by hackers and criminals in order to remotely control hacked

computers. For example, researchers at Kaspersky Lab have published research uncovering three different campaigns that use Tor as a host infrastructure for criminal malware activities: a *64-bit version of the Zeus Trojan* [11] that sends traffic through Tor and creates Tor hidden services to obscure the hackers' location; *Chewbacca* [12], a Trojan that steals data from the memory of compromised computers, and communicates over Tor; and most recently an Android Trojan that uses a .onion domain as a command and control infrastructure.

In this paper we present our methodology for detecting any connection to or from Tor network. Our detection method is based on a list of Tor servers. We process the network traffic and match the source and destination IP addresses for each connection with Tor servers list. The list of Tor servers is automatically updated each day and the detection is in the real time. Tor servers list is not generally effective at detecting new or previously unknown Tor servers because the Tor server has to be known before they can be added to the list. However, as part of a larger solution, performing listing of Tor servers is generally worth the effort. Depending on Tor servers list is relatively "low cost" in that using them in blocking rules or log searches doesn't severely impact system performance.

This detection method is one of other detection methods (like malicious domain name detection, domain flux detection, disguised exe file detection and spear phishing email detection) for detecting other techniques used in advanced persistent threat (APT) attack life cycle [13][14]. The output of this detection method will be correlated with the outputs of other detection methods to raise an alert on APT attack detection. In APT attack, Tor connection can be used to communicate with the Command and Control servers or to exfiltrate the stolen data.

We applied our methodology on campus live traffic and showed that it can automatically detect Tor connections in the real time. We also applied our methodology on packet capture (pcap) files which contain real and long-lived malware traffic and we proved that our methodology can successfully detect Tor connections.

The remainder of this paper is organized as follows. Section 2 presents previous related work to Tor connection detection. Our methodology and implementation of our detection method are explained in Section 3. Section 4 shows our results and section 5 concludes the paper.

2 Related work

In [15], they designed an "early warning" system that looks for anomalies in the volumes of connections from users in different jurisdictions and flags potential censorship events. Special care had been taken to ensure the detector is robust to manipulations and noise that could be used to block without raising an alert. The detector works on aggregate number of users connecting to a fraction of directory servers per day.

In [16], they explored the use of decoys at multiple levels for the detection of traffic interception by malicious nodes of proxy-based anonymous communication systems. Their approach relies on the injection of traffic that exposes bait credentials for decoy services requiring user authentication, and URLs to seemingly sensitive decoy documents which, when opened, invoke scripts alerting about being accessed. Their aim was to entice prospective eavesdroppers to access their decoy servers and decoy documents, using the snooped credentials and URLs.

A novel use of bridge relays to provide persistent Tor connections for mobile devices was proposed in [17], they assess the potential performance impact to a mobile user accessing Tor while roaming from different Internet connections. An experiment was undertaken to simulate a mobile user at various mobility speeds (e.g. walking) alongside a range of Tor circuit build times.

In [18], they focused on the detection of bridges that are used in a controlled network. Their detecting methods rely on two observations. (1) Since only partial bridges are published in a certain period, the same bridges will occur repeatedly, resulting in high correlation among tuples. (2) When Tor clients try to connect chosen hidden bridges, multiple SYN packets with consecutive source ports will be sent almost simultaneously, destining for different hosts. If any destination IP contained among such packets belongs to the current known bridge set, all others can then be inferred to be of bridges too. Therefore, under the assumption that they had got an initial bridge set, the bridges used in a controlled network can be effectively detected by monitoring the traffic pattern and correlating suspected bridge tuples with known bridges.

New countermeasure of cyber attacks in mix networks was presented in [19], they verified statistically the attacking methods which the hackers shall create by increasing the data transmission rate of TOR through manipulating the speed of the anonymous network, and proposed political countermeasures to detect effectively hacker's attacks which use Tor network.

With regards to Tor hidden services, the first published attacks against Tor hidden services were presented in [20]. They targeted a previous version of the hidden services design in which no entry guard nodes were used. In the scenario described, the attacker needs to control one or more Tor relays; the idea being that given enough connection attempts, one of the attacker's relays will be chosen as the first hop of the rendezvous circuit established by the hidden service.

In [21], they exposed flaws both in the design and implementation of Tor's hidden services that allow an attacker to measure the popularity of arbitrary hidden services, take down hidden services and deanonymize hidden services. They gave a practical evaluation of their techniques by studying: (1) a recent case of a botnet using Tor hidden

services for command and control channels; (2) Silk Road, a hidden service used to sell drugs and other contraband; (3) the hidden service of the DuckDuckGo search engine.

3 Methodology

In this section we propose our methodology for detecting any connection to or from Tor network. Our detection method is based on a list of Tor servers. As it is shown in Figure 1, we process the network traffic and match the source and destination IP addresses for each connection with Tor servers list. The list of Tor servers is automatically updated each day and the detection is in the real time.

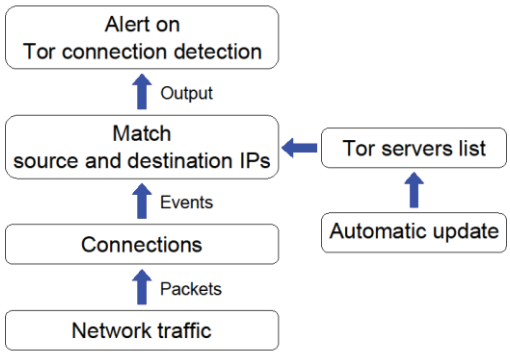

Figure 1: Methodology of Tor connection detection

We have implemented our detection method on top of Bro [22]. Bro is a passive, open-source network traffic analyzer. It is primarily a security monitor that inspects all traffic on a link in depth for signs of suspicious activity. The most immediate benefit that we gain from deploying Bro is an extensive set of *log files* that record a network's activity in high-level terms. These logs include not only a comprehensive record of every connection seen on the wire, but also application-layer transcripts such as, e.g., all HTTP sessions with their requested URIs, key headers, MIME types, and server responses; DNS requests with replies; and much more.

Figure 2 explains the implementation of our detection method in Bro, first we start processing the network traffic. Bro is able to reduce the incoming packet stream into a series of higher-level *events*, so we can get *connection_established* event. This event is generated when a SYN-ACK packet is seen in response to a SYN packet during a TCP handshake. Through this event we should check both connection sides to detect if the connection is to or from Tor network.

In the first case, detecting if the connection is to Tor network, first we check if this connection is established by a host from our network, therefore we check the source IP address through *is_local_addr* function, this function returns true if an address corresponds to one of the defined local networks, false if not. After that, we check if the destination IP address is in *t_tor_server* table, this table contains IP addresses of Tor servers which are publicly published [23], and it is automatically updated each day as it is shown in Figure 3. When a match is found, that means that the connection is established to TOR network but before we raise an alert we check if the source IP address is not exist in *t_suppress_tor_alert* table, this table is to suppress the alerts to one alert about the same IP address (same Tor client) per day. If the source IP address is not exist in *t_suppress_tor_alert* table, we should add it to the table and it will stay there for one day to be sure that we will not get another alert about the same IP address during one day.

Now we can write the following information into *tor_detection.log*:
timestamp = c$start_time
alert_type = " tor_alert "
connection = c$id
infected_host = cidorig_h
tor_server = cidresp_h

We send an alert email about Tor connection detection to RT (Request Tracker) where the network security team can perform additional forensics and response to it [24]. As we have mentioned in section 1, this detection method is one of other detection methods for detecting other techniques used in advanced persistent threat (APT) attack life cycle. The output of this detection method will be correlated with the outputs of other detection methods to raise an alert on APT attack detection; therefore we generate an event about Tor connection detection to be used for alert correlation [25].

In the second case, detecting if the connection is from Tor network, we follow the same procedure of the first case but we pay attention to the source and destination IP addresses as it is explained in Figure 2.

For automatic update of *t_tor_server* table which is used in our methodology, Figure 3 shows how it is done.

Figure 3: Automatic update for Tor servers list

We start from user *crontab file* which is configured to run *blacklist_update.sh* each day at 3:00 am, this shell script will connect through Internet to the data source website (public Tor servers) and download updated list of Tor servers into *tor_server_list.txt* file. Then we have Input Framework which enables us to use the text file as an input for our methodology which is implemented on top of Bro, this framework will read *tor_server_list.txt* file into *t_tor_server* table and this table will be used by event handler as it is explained above. This automatic update is done without stopping network traffic live monitor.

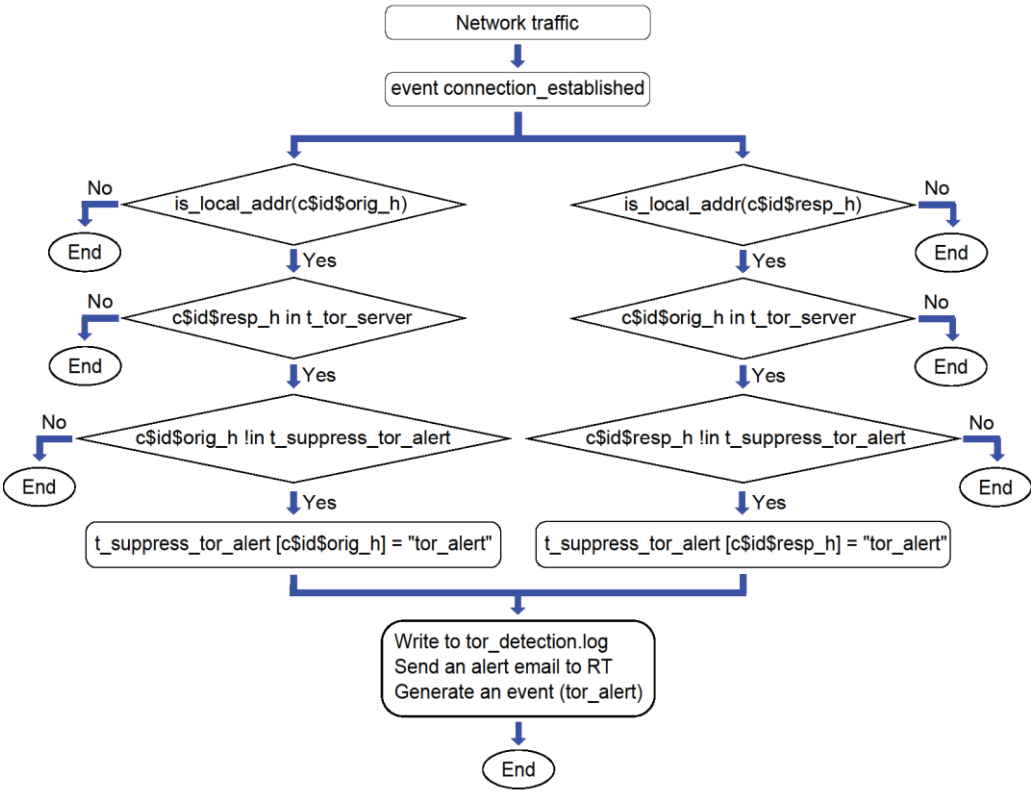

Figure 2: Implementation of Tor connection detection method

4 Evaluation and results

We used four scenarios to evaluate our methodology. In the first scenario, we ran Tor Browser Bundle [26] on a computer connected to the monitored network. In the second scenario, we observed the campus network for hosts using Tor connections. A third-party pcap file with a trace of malware communication was analyzed in the third scenario. In the fourth scenario, we set up our own virtual network with a live malware sample that used Tor connections.

In the first scenario, Tor Browser Bundle was installed on a computer connected to the Internet through a network monitored using our detection method. We focused on the real-time detection capabilities of our method in this scenario. The method was set up to send a report to RT (Request Tracker) as soon as a Tor connection was detected. Request Tracker is often used to help incident handlers to deal with events that need an automated action or human attention.

The test consisted of the following steps. First, Tor Browser Bundle was started. We noted the start-up time with second precision. Second, a random web page was loaded in Tor Browser Bundle as soon as the browser started. Because all network requests from Tor Browser Bundle are routed through the Tor network, the selection of loaded web sites makes no difference for the test. The detection method detected a Tor connection after the first step and automatically created an RT ticket. This Tor connection was from our computer based on the source IP address of the connection. Third, we received the RT ticket. We compared the start-up time with the time in the RT ticket and noted the detection delay. Since the minimum supported time resolution is 1 second, the test was repeated 25 times and the time difference was measured. Figure 4 shows the results.

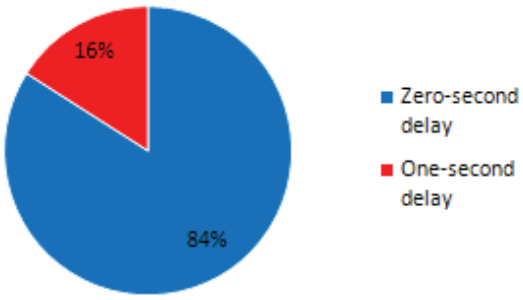

Figure 4: Observed delays of our detection method

In the second scenario, we monitored the campus live traffic for Tor connections. The method was set up to create a log file of detected Tor connections. We set up a server hosting our detection method and passively analyzing the campus live traffic. The analysis was performed for one month. We correlated the list of hosts communicating over Tor with results of a domain flux detection method [27]. As it is shown in Figure 5, 24 hosts were detected to use Tor and 31 hosts were detected to use domain flux. Of these, 13 hosts were detected to use both Tor and domain flux, which indicated they were infected with malware.

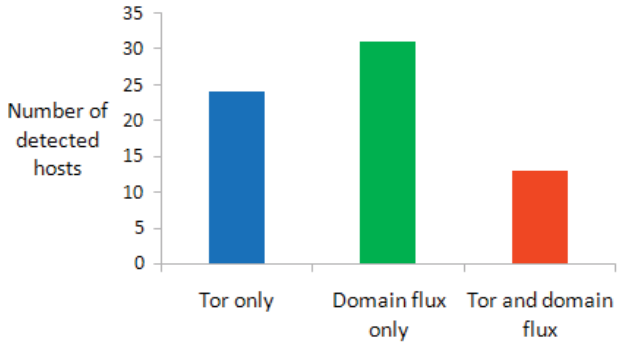

Figure 5: Machines detected to use Tor and domain flux

We applied our methodology on pcap files in the third scenario. The pcap files were created by Contagio and contained traffic created by the Trojan.Tbot (Skynet Tor Botnet) malware [10]. The Trojan.Tbot malware uses Tor network to communicate with its C&C center. The provided pcap files contain Tor traffic initiated by the malware. We used this fact to set the ground truth. We set up our method so that it consumed pcap files and produced log files. We applied our detection method on the provided pcap files and it was successfully able to detect all the Tor connections in these pcap files. Figure 6 shows part of *tor_detection.log* produced by our detection method.

```
#fields timestamp  alert_type infected_host  tor_server
#types  time       string     addr           addr
1349621090.423721 tor_alert 172.16.253.130 208.83.223.34
1349621090.945925 tor_alert 172.16.253.130 86.59.21.38
1349621102.634850 tor_alert 172.16.253.130 74.120.13.132
1349621102.628802 tor_alert 172.16.253.130 96.47.226.20
1349621102.670488 tor_alert 172.16.253.130 96.44.189.102
1349621102.728990 tor_alert 172.16.253.130 85.10.237.104
1349621102.741374 tor_alert 172.16.253.130 31.172.30.4
1349621102.750483 tor_alert 172.16.253.130 31.172.30.3
1349621102.750386 tor_alert 172.16.253.130 5.9.191.52
1349621102.815528 tor_alert 172.16.253.130 78.108.63.44
```

Figure 6: Part of the log produced by our detection method

The fourth scenario extended the approach from the previous scenario. We set up a test network with connectivity to the Internet, installed a live malware sample, recorded the traffic to pcap files, and used our detection method to analyze the pcap files. The malware we analyzed was Trojan.Win32.Scarsi.uhm with MD5 hash 52d3b26a03495d02414e621ee4d0c04e [28]. The malware uses Tor to connect to the C&C center. As it is shown in Figure 7, two Windows XP SP3 virtual machines were connected to a physical consumer-grade router. The virtual machines behaved as physical computers in a home network and were able to communicate between each other. Moreover, the router provided connection to the Internet. Virtual machines traffic was recorded to two pcap files using the nictrace VirtualBox functionality [29], one pcap file per virtual machine. Because no software besides the operating system and malware was installed on the virtual machines and operating system updates were disabled, it was possible to conclude that the majority of the virtual machine traffic was initiated by the installed malware.

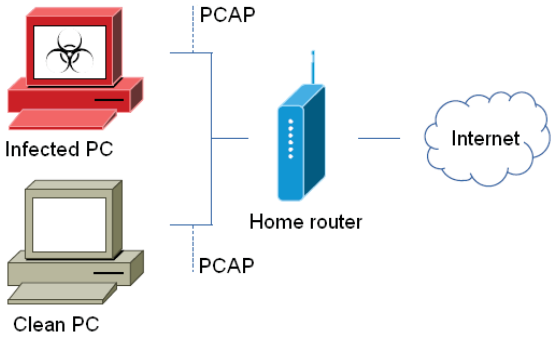

Figure 7: Topology of the test network

We ran the malware on one virtual machine for 10 hours. Through manual traffic analysis, we established that the first virtual machine connected to Tor and the second virtual machine didn't connect to Tor and it didn't become infected. Tor connection was inferred from observed SSL connections to IP addresses belonging to the Tor network based on a public IP list. The infected virtual machine showed a surge of communication in the first two minutes after launch of the malware executable. In this time period, it transferred 3698 kB of data, and initiated 67 connections to 67 unique addresses (first hops of the Tor connections) in 12 countries. Figure 8 shows countries of the first hops during the first two minutes. After that, the flow of traffic dropped but variability of contacted addresses remained. We used the manual analysis to establish the ground truth.

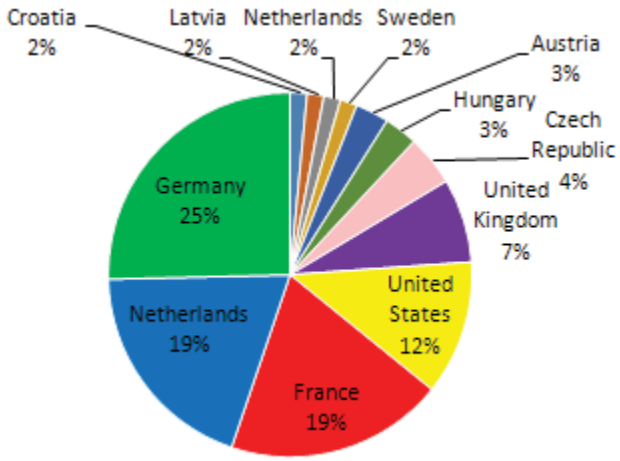

Figure 8: Locations of the first hops used by the malware

We then applied our detection method on the pcap files of the virtual machines traffic. Our detection method was able to detect all Tor connections which we previously detected manually. Traffic of the infected virtual machine triggered the detection, while the traffic of the clean virtual machine didn't trigger the detection. The number of different countries contacted in the malware communication is radically different from a user-initiated session using the Tor Browser Bundle, which uses only one address for the first hop as long as possible. This difference can be further used for better malware detection in the following work.

5 Conclusion

In this paper we present our methodology for detecting any connection to or from Tor network. Our detection method is based on a list of Tor servers. We applied our methodology on campus live traffic and showed that it can automatically detect Tor connections in the real time. We also applied our methodology on packet capture (pcap) files which contain real and long-lived malware traffic and we proved that our methodology can successfully detect Tor connections. For future work, the output of our detection method will be correlated with the outputs of other detection methods to raise an alert on APT attack detection.

Acknowledgements

This work has been supported by the project "CYBER-2" funded by the Ministry of Defence of the Czech Republic under contract No. 1201 4 7110.

References

[1] P. Winter and S. Lindskog, "How the great firewall of china is blocking tor," FOCI. USENIX Association, 2012.

[2] T. Elahi and I. Goldberg, "Cordon–a taxonomy of internet censorship resistance strategies," University of Waterloo CACR, vol. 33, 2012.

[3] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The secondgeneration onion router," DTIC Document, Tech. Rep., 2004.

[4] "Tor metrics," https://metrics.torproject.org/, accessed: 1-10-2014.

[5] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding routing information," in Information Hiding. Springer, 1996, pp. 137–150.

[6] M. G. Reed, P. F. Syverson, and D. M. Goldschlag, "Anonymous connections and onion routing," Selected Areas in Communications, IEEE Journal on, vol. 16, no. 4, pp. 482–494, 1998.

[7] D. L. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," Communications of the ACM, vol. 24, no. 2, pp. 84–90, 1981.

[8] S. Doswell, N. Aslam, D. Kendall, and G. Sexton, "Please slow down!: the impact on tor performance from mobility," in Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices. ACM, 2013, pp. 87–92.

[9] W. Diffie and M. E. Hellman, "New directions in cryptography," Information Theory, IEEE Transactions on, vol. 22, no. 6, pp. 644– 654, 1976.

[10] NETRESEC, "Detecting tor communication in network traffic,"http://www.netresec.com/?page=Blog&month=2013-04&post=Detecting-TOR-Communication-in-Network-Traffic, accessed: 1-10-2014.

[11] "The inevitable move - 64-bit zeus enhanced with tor," http://securelist.com/blog/events/58184/the-inevitable-move-64-bit-zeus-enhanced-with-tor/, accessed: 1-10-2014.

[12] "Chewbacca latest malware to take a liking to Tor," http://threatpost.com/chewbacca-latest-malware-to-take-a-liking-to-tor/103220, accessed: 1-10-20

[13] N. Virvilis and D. Gritzalis, "The big four-what we did wrong in advanced persistent threat detection?". Eighth International Conference on. IEEE, 2013, pp. 248–254.

[14] D. SECUREWORKS, "Lifecycle of the advanced persistent threat," cit, pp. 05–11, 2013.

[15] G. Danezis, "An anomaly-based censorship detection system for tor," The Tor Project, 2011.

[16] S. Chakravarty, G. Portokalidis, M. Polychronakis, and A. D. Keromytis, "Detection and analysis of eavesdropping in anonymous communication networks," International Journal of Information Security, pp. 1–16, 2014.

[17] S. Doswell, N. Aslam, D. Kendall, and G. Sexton, "The novel use of bridge relays to provide persistent tor connections for mobile devices," in Personal Indoor and Mobile Radio Communications (PIMRC), 2013 IEEE 24th International Symposium on. IEEE, 2013, pp. 3371–3375.

[18] M. Yang, J. Luo, L. Zhang, X. Wang, and X. Fu, "How to block tors hidden bridges: detecting methods and countermeasures," The Journal of Supercomputing, vol. 66, no. 3, pp. 1285–1305, 2013.

[19] K. C. Park, H. Shin, W. H. Park, and J. I. Lim, "New detection method and countermeasure of cyber attacks in mix networks," Multimedia Tools and Applications, pp. 1–10, 2014.

[20] L. Overlier and P. Syverson, "Locating hidden servers," in Security and Privacy, 2006 IEEE Symposium on. IEEE, 2006, pp. 15–pp.

[21] A. Biryukov, I. Pustogarov, and R. Weinmann, "Trawling for tor hidden services: Detection measurement, deanonymization," in Security and Privacy (SP), 2013 IEEE Symposium on. IEEE, 2013, pp. 80–94.

[22] Bro project, "The bro network security monitor", http://www.bro.org/, accessed: 1-10-2014.

[23] J. B. Kowalski, "Tor status network", http://torstatus.blutmagie.de/ip_list_all.php/Tor_ip_list_ALL.csv, accessed: 1-10-2014.

[24] "Rt: Request tracker", https://www.bestpractical.com/rt/, accessed: 1-10-2014.

[25] Wikipedia, "Alert correlation", http://en.wikipedia.org/wiki/Alertcorrelation, accessed: 1-10-2014.

[26] "Tor browser", https://www.torproject.org/projects/torbrowser.html.en, accessed: 1-10-2014.

[27] Wikipedia, "Domain generation algorithm", http://en.wikipedia.org/wiki/Domain_generation_algorithm, accessed: 1-10-2014.

[28] NETRESEC, "Trojan.win32.scarsi.uhm", https://malwr.com/analysis/MDc1NDJkNjRhYjRhNGNiZjg2MGNkNGMzN2JiZjZjNTQ/, accessed: 1-10-2014.

[29] "Network tracing", https://www.virtualbox.org/wiki/Network_tips, accessed: 1-10-2014.