TorWard: Discovery, Blocking, and Traceback of Malicious Traffic Over Tor

Zhen Ling, Junzhou Luo, *Member, IEEE*, Kui Wu, *Senior Member, IEEE*, Wei Yu, and Xinwen Fu

***Abstract*— Tor is a popular low-latency anonymous communication system. It is, however, currently abused in various ways. Tor exit routers are frequently troubled by administrative and legal complaints. To gain an insight into such abuse, we designed and implemented a novel system, TorWard, for the discovery and the systematic study of malicious traffic over Tor. The system can avoid legal and administrative complaints, and allows the investigation to be performed in a sensitive environment such as a university campus. An intrusion detection system (IDS) is used to discover and classify malicious traffic. We performed comprehensive analysis and extensive real-world experiments to validate the feasibility and the effectiveness of TorWard. Our results show that around 10% Tor traffic can trigger IDS alerts. Malicious traffic includes P2P traffic, malware traffic (e.g., botnet traffic), denial-of-service attack traffic, spam, and others. Around 200 known malwares have been identified. To mitigate the abuse of Tor, we implemented a defense system, which processes IDS alerts, tears down, and blocks suspect connections. To facilitate forensic traceback of malicious traffic, we implemented a dual-tone multi-frequency signaling-based approach to correlate botnet traffic at Tor entry routers and that at exit routers. We carried out theoretical analysis and extensive real-world experiments to validate the feasibility and the effectiveness of TorWard for discovery, blocking, and traceback of malicious traffic.**

***Index Terms*— Tor, malicious traffic, intrusion detection system.**

Manuscript received January 18, 2015; revised May 31, 2015 and July 22, 2015; accepted July 22, 2015. Date of publication August 7, 2015; date of current version September 25, 2015. This work was supported in part by the China National High Technology Research and Development Program under Grant 2013AA013503, in part by the National Natural Science Foundation of China under Grant 61502100, Grant 61572130, Grant 61532013, Grant 61502098, Grant 61502099, Grant 61272054, Grant 61202449, Grant 61402104, and Grant 61320106007, in part by a Discovery Grant (195819339) from the Natural Sciences and Engineering Research Council of Canada, in part by the U.S. National Science Foundation under Grant 1461060, Grant 1116644, Grant 1350145, and Grant CNS 1117175, in part by Jiangsu Provincial Natural Science Foundation of China under Grant BK20150637, Grant BK20150628, and Grant BK20150629, in part by Jiangsu Provincial Key Technology R&D Program under Grant BE2014603, in part by Jiangsu Provincial Key Laboratory of Network and Information Security under Grant BM2003201, and in part by Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grant 93K-9. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Wanlei Zhou.

Z. Ling and J. Luo are with the School of Computer Science and Engineering, Southeast University, Nanjing 210096, China (e-mail: zhenling@seu.edu.cn; jluo@seu.edu.cn).

K. Wu is with the Department of Computer Science, University of Victoria, Victoria, BC V8W 3P6, Canada (e-mail: wkui@cs.uvic.ca).

W. Yu is with the Department of Computer and Information Sciences, Towson University, Towson, MD 21252 USA (e-mail: wyu@towson.edu).

X. Fu is with the Department of Computer Science, University of Massachusetts Lowell, Lowell, MA 01854 USA (e-mail: xinwenfu@cs.uml.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2015.2465934

I. Introduction

TOR IS a popular overlay network that provides anonymous communication over the Internet for TCP applications and helps fight against various Internet censorship [1]. It serves hundreds of thousands of users and carries terabyte of traffic daily. Unfortunately, Tor has been abused in various ways. Copyrighted materials are shared through Tor. The black markets (e.g., Silk Road [2], an online market selling goods such as pornography, narcotics or weapons[1]) can be deployed through Tor *hidden service*. Attackers also run botnet Command and Control (C&C) servers and send spam over Tor.

Attackers choose Tor because of its protection of communication privacy, which is achieved in the following way. A user uses source routing, selects a few (3 by default while the hidden service uses a different mechanism [3]) Tor routers, and builds an anonymous route along these Tor routers. Traffic between the user and the destination is relayed along this route. The last hop, called *exit router*, acts as a "proxy" to directly communicate with the destination. Hence, Tor exit routers often become scapegoats and are bombarded with Digital Millennium Copyright Act (DMCA) notices and botnet and spam complaints. In some cases, they are even raided by police [4]. Since Tor exit routers are mainly hosted by volunteers, these abusing activities prevent potential volunteers from hosting exit routers and hinder the advancement of Tor as a large-scale privacy-enhancing network.

Tor allows manual configuration of IP and port based policies to block potential malicious traffic. However, traffic over Tor has versatile ports such as P2P traffic, making manual configuration a daunting job for common Tor router administrators. Hence, a pressing need is to investigate malicious traffic over Tor. Our research in this paper fills this gap and differs from the existing research efforts, which mainly focus on traffic protocols and applications. For example, McCoy *et al*. [5] reported that web traffic made up the majority of the connections and bandwidth in 2008. Chaabane *et al*. [6] conducted the analysis of the application usage over Tor through deep packet inspection and found that BitTorrent became the first contributor in terms of traffic volume in 2010.

In this paper, we design and implement *TorWard*, which integrates an Intrusion Detection System (IDS) at Tor exit routers for Tor malicious traffic discovery, classification and response. An early version of TorWard [7] can discover and classify malicious traffic in Tor while the new TorWard

[1] On Oct. 2 2013, the FBI took down Silk Road.

introduced in this paper can also block and track malicious traffic.

First, *TorWard* can be deployed in a sensitive environment such as *a university campus* without causing legal and administrative complaints. It consists of a NAT (Network Address Translation) gateway and a Tor exit router behind the gateway. Tor traffic is routed through the gateway to the exit router so that we can study the outgoing traffic from Tor. The traffic leaving our exit router is redirected into Tor again through the gateway to relieve the university from legal liability. We understand rerouting exit traffic into Tor incurs a burden on Tor. Nonetheless, this is the only safe way to investigate malicious traffic over Tor in a sensitive environment. An IDS is installed on the NAT gateway to analyze the exit traffic before it is rerouted into Tor. We revise the Tor source code and dynamically maintain firewall rules to avoid interference with non-Tor traffic. Since the use of *TorWard* in early 2012, we have not received any legal and administrative warning, while a lot of administrative complaints were received each day with a bare exit router on campus.

Second, with *TorWard*, we carry out statistical analysis of malicious traffic through Tor. A key observation is that around 10% of Tor traffic triggers the IDS alerts. Alerts are very diverse, raised over botnet traffic, DoS attack traffic, spam traffic, and others. More than 200 malware are discovered from the alerts, including 5 mobile malware, all targeting Android. Although we did not manually filter out all false alarms given the huge volume of traffic, we demonstrate examples of major threats such as botnet traffic. Our goal of this paper is to show the pressure of malicious traffic over Tor exits and draw a baseline for future intrusion detection classification and analysis. In addition, we derive *traffic protocol and application* statistics, which is largely consistent with the study in [5] and [6] while we now can observe traffic from mobile devices. To the best of our knowledge, this is the first effort to categorize malicious traffic over Tor.

Third, to mitigate the abuse of Tor, we design and implement a defense mechanism to block malicious traffic. The IDS is configured to send alerts to a *sentinel* agent, which retrieves the source IP addresses and ports of the suspicious traffic and sends the tear-down command to the exit router through our modified Tor control protocol. The exit router disconnects the specific connection. We deploy the IDS with extensive rule sets, which are updated regularly to block as much malicious traffic as possible. Although IDS may produce false alarms, the problem is outweighed by the benefit when more people become willing to host Tor exit routers without concerning administrative and legal issues.

Fourth, *TorWard* can be used to trace back malicious traffic across Tor for forensic purpose. As an example that itself has significance, we trace IRC botnet traffic, which composes the majority of observed botnet traffic. We propose a dual-tone multi-frequency (DTMF) signaling based approach to correlate botnet traffic at Tor entries and that at Tor exits. We use "phantom" IRC messages, which do not interfere with the bot or botmaster, and pack these IRC messages into cells and control the cell transmission frequencies at our exit router.

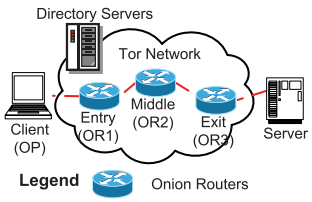

Fig. 1. Tor network.

Two frequencies are used to embed secret binary signals into a target circuit. Once the two *feature frequencies* are detected at our controlled entry routers, the suspect botnet IP address can be identified. Administrative and legal procedures can be taken against the suspect IP. We carry out theoretical analysis and extensive experiments and our data shows that the DTMF based traceback is simple, efficient (no need of injecting much traffic), and effective.

The rest of this paper is organized as follows. We introduce Tor and review related work in Section II. We present the system architecture for malicious traffic discovery, system setup, and theoretical analysis to demonstrate the effectiveness of *TorWard* in Section III. We conduct a statistical analysis on Tor traffic and investigate various alerts and malware activities in Section IV. In Section V, we present approaches to blocking and tracing malicious traffic. In Section VI, we analyze the effectiveness of the traceback approach. In Section VII, we show experimental results and validate our findings. We conclude this paper in Section VIII.

II. Background and Related Work

A. Tor

Figure 1 illustrates the basic architecture of the Tor network. It consists of four components: Tor clients, onion routers, directory servers, and application servers. Generally speaking, a Tor client installs *onion proxy* (*OP*), which is an interface between Tor network and clients. *Onion routers* (*OR*) form the core Tor network and relay traffic between a Tor client and an application server. The directory servers hold all public onion router information. An application server hosts a TCP application service such as a web. Tor also provides a *hidden service* to hide the location of servers. *Bridge* is introduced as hidden onion routers to further resist censorship. Without loss of generality, we will use Figure 1 as the example architecture of the Tor network in this paper.

To anonymously communicate with the remote server over Tor, the client downloads onion router information from a directory server and chooses a series of onion routers to establish a three-hop path, referred to as *circuit*. The three onion routers are known as *entry* (OR1), *middle* (OR2), and *exit* onion router (OR3), respectively. When a circuit is created by the client, the client (i.e., OP) negotiates three distinct Diffie-Hellman (DH) keys with these three routers - entry, middle, and exit routers, respectively. To transmit data to a remote server, the client packs the data into basic transmission units, referred to as *cells*, where each cell has a fixed size of 512 bytes. The cell will be encrypted in the onion-like fashion using these three DH keys stored at the client side. The encrypted cell will be transmitted and decrypted at each router in the onion-like fashion. Finally, the exit router has

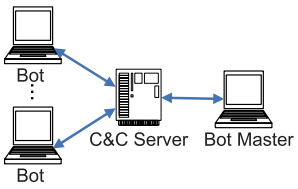

Fig. 2. Centralized botnet.

the fully decrypted data and deliver it to the remove server. Therefore, the client data is encrypted while being transmitted inside the Tor network. An IDS can only inspect the content of the client data at the exit router, but not at an entry or middle router.

B. Botnet

Botnets have become a primary platform to launch various Internet attacks, such as spam, distributed denial-of-service (DDoS), identify theft and phishing. There are two typical botnet structures: P2P botnet and centralized botnet. In the P2P botnet, bots can form an overlay network and each bot can be controlled by the botmaster to distribute commands to the other peers or collect information from them. Although the P2P botnet is more complicated and probably more costly to manage, this structure offers higher resiliency. In the centralized botnet, protocols such as IRC and HTTP are used to build a centralized command and control (C&C) architecture. The centralized C&C channels have been widely used by many botnets because of the simplicity and availability of open source and reusable C&C server code.

Figure 2 illustrates the architecture of a common centralized botnet C&C. As we can see, there are three components: (i) *Botmaster*, which relies on the C&C channel to issue commands to their bots and receives information from infected hosts through the C&C server, (ii) *C&C server*, which relays the commands to bots on behalf of the botmaster, collects information from bots, or forwards results to the botmaster, and (iii) *Bots*, which are compromised computers with botware and receive control commands from the C&C server to actually commit attacks.

C. Related Work

The most related work [5], [6] focuses on the network protocol analysis to study the *benign* use of Tor. In comparison, our work explores malicious traffic over Tor and proposes countermeasures.

Malware authors have used Tor to provide anonymous communication between malware and C&C servers. For example, Brown [8] showed how to configure Zeus bot through Tor2Web [9] to connect to its C&C server, which is deployed as a Tor hidden server. Malware Skynet was discussed in the web site Reddit in 2012 [10]. It deploys the C&C server as a hidden server and embeds a Tor client into the malware to communicate with the hidden C&C server. Guarnieri studied the detailed features of Skynet by dissecting malware samples [11], [12]. Two malwares using Tor hidden service [13], [14] were reported in July 2013.

Research has been performed to discover Tor hidden servers [3], [15]–[18]. For example,

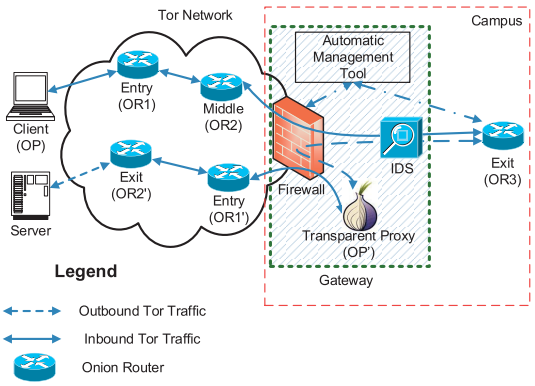

Fig. 3. System architecture for malicious traffic collection.

Øverlier and Syverson [15] proposed a traffic analysis method based on packet counting to identify a hidden server at entry onion routers. Zhang *et al.* [17] leveraged HTTP features to identify a hidden server at entry onion routers. Murdoch [16] employed an approach using clock skew to check whether or not a given Tor node is a hidden server. Ling *et al.* [3] proposed a protocol-level approach for discovering hidden servers. Biryukov *et al.* [18] studied the method of deploying the hidden service directory to harvest hidden service information and investigated the traffic analysis method based on packet counting to locate hidden servers.

Like Tor, other anonymous communication systems were widely abused. For example, Tian *et al.* [19] studied how to trace back the receiver who is retrieving illegal file over the Freenet [20]. Xiang et al. [21] and Yu *et al.* [22] investigated the possible traceback techniques and countermeasures.

III. System Architecture of TorWard

In this section, we first present the architecture design of *TorWard* to collect and analyze malicious traffic in the live Tor network and then elaborate the detailed system setup. At last, we analyze the effectiveness of *TorWard*.

A. System Architecture

Tor traffic can be classified as inbound and outbound traffic. Inbound Tor traffic is encrypted and transmitted between OR and OR or between OP and OR. Outbound Tor traffic is decrypted by the Tor exit router and forwarded to an application server. An exit router behaves as a proxy for a Tor client and communicates with the application server. Hence, media companies, ISPs (Internet Service Providers), and campus IT department may detect malicious outbound Tor traffic and direct complaints to "offending" exit router administrators.

It is nearly impossible to study malicious activities over Tor on campus because of continuous administrative and legal complaints. We design *TorWard* to address this challenge. Figure 3 illustrates the structure of this system. *TorWard* consists of four logical components: a firewall, an IDS, a transparent proxy, and a Tor exit router. Port forwarding is enabled at the firewall to enable communication between the exit router and middle routers in the public network. To attract

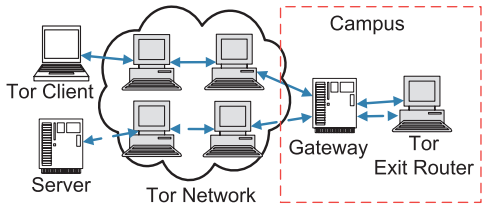

Fig. 4. Experiment setup for malicious traffic discovery.

other Tor clients to select our exit router, our exit router is set to accept all traffic and has a relatively large average bandwidth and burst bandwidth of $16Mbps$ and $32Mbps$, respectively.

To avoid administrative and legal complaints, *TorWard* redirects outbound traffic at our exit router into the Tor network. We develop an automatic management tool to automatically add and delete forwarding rules for the firewall. We modify the code of the exit router in order to send the outbound connection information (i.e., the destination IP address and port) to this tool. In particular, before an exit router initiates an outbound connection, we send the connection's destination IP address and port to this tool and add a rule for the connection. When the Tor exit router closes the outbound connection, the tool is informed to remove the corresponding rule from the firewall. The Tor client is configured to act as a transparent proxy [23] and listens on port 9040. The rules added by our tool actually redirect corresponding outbound Tor connections to this proxy port. This procedure is completely transparent to the Tor exit router. To improve the performance, we modify the client code to establish a two-hop circuit and relay the outbound Tor traffic into the Tor network. In this way, the remote real Tor clients actually use five-hop circuits.

B. System Setup

Figure 4 shows the system setup of *TorWard*. We use one computer with two network interfaces as a gateway connected to our private network on campus. Another computer connects to the gateway as a Tor exit router. A firewall, an IDS, and a transparent proxy are hosted at a NAT gateway machine. We built a private network and set up a firewall on the gateway, which connects the private network to the campus network. The private network includes a computer that works as a Tor exit router. Both computers use Fedora Core 15. As stated in Section III-A, we enable NAT and port forwarding at the firewall through *iptables* on the gateway. A Tor client is installed at the gateway and is configured as a transparent proxy. Our automatic firewall rule management tool is deployed on the gateway and dynamically maintains firewall rules to redirect outbound Tor traffic from the exit router to the transparent proxy. The code of Tor exit router is modified to send the connection information to the automatic firewall rule management tool. An IDS, *Suricata* [24], is deployed on the gateway. Notice that any IDS can be used as the IDS component in TorWard. Since Suricata is one of the well-known open source IDSs, we use Suricata as an example to demonstrate how TorWard works. The IDS detects the destination of outbound traffic from the Tor exit router and uses signature-based rules to detect potential malicious traffic.

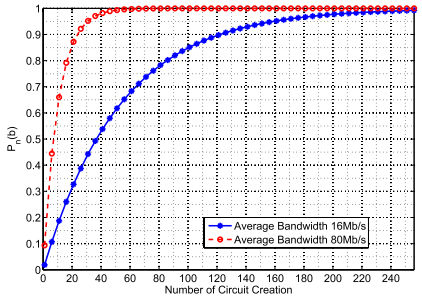

Fig. 5. P_n versus number of circuits.

We adopt the IDS rules from Emerging Threats [25] for *Suricata*. It is worth noting that *Suricata* records alerts into unified binary files, and *Barnyard2* [26] is configured to read the alerts stored in the unified binary files and sends alerts to a *MySQL* database. The *MySQL* database is installed on the gateway computer and *Barnyard2* acts as a bridge to promptly send the alerts from *Suricata* to *MySQL*. In addition, *BASE* [27] is deployed as a GUI to display alerts stored in the database.

C. Effectiveness of TorWard

To demonstrate the effectiveness of *TorWard* and confirm the hypothesis that we can use a few or even one exit router to derive reliable traffic statistics over Tor, we conduct theoretical analysis to derive the probability that the malicious traffic traverses our deployed Tor exit router. We assume that there are m distinct types of malicious clients and each client generates different malicious traffic through Tor. A Tor client will create a three-hop circuit to relay malicious traffic to the Tor exit router and the exit router will forward the traffic to the real destination. If our exit router is selected by the malicious Tor client, *TorWard* can detect the malicious traffic. Consequently, we need to determine the probability $\mathcal{P}$ that the malicious Tor client selects our Tor router as the exit router in its circuits. According to the Tor weight bandwidth path selection algorithm [28], the probability $\mathcal{P}$ can be derived by the proportion of the bandwidth of our Tor exit router and the total weighted bandwidth of Tor exit routers. Tor routers can be categorized into four groups: pure entry routers, pure exit routers, both entry routers and exit routers, and neither entry routers nor exit routers, whose total bandwidth is denoted as $\mathcal{B}$, $\mathcal{B}_e$, $\mathcal{B}_x$, and $\mathcal{B}_{ee}$ respectively. Let b be the bandwidth of our Tor exit router. Then, the probability that the malicious Tor client selects our Tor router as the exit router in their circuits can be calculated with (1),

$$\mathcal{P}(b) = \frac{b}{\mathcal{B}_x + \mathcal{B}_{ee} * w}, \tag{1}$$

where the weight w is equal to $max\{0, 1 - \frac{\mathcal{B}}{3(\mathcal{B}_e+\mathcal{B}_{ee})}\}$.

Assume a malicious Tor client builds several circuits to send malicious traffic through Tor. Denote n as the number of circuits. After creating n circuits, the probability that at least one circuit traverse our exit router, denoted as $\mathcal{P}_n$, is

$$\mathcal{P}_n(b) = 1 - (1 - \mathcal{P}(b))^n. \tag{2}$$

Hence, we can see that $\mathcal{P}_n$ grows significantly as n increases. According to the current Tor router bandwidth real-world

data [29], we can calculate the probability $\mathcal{P}_n(b)$ based on the number of circuits n. We set up the bandwidth of our exit router as $16Mb/s$, while the theoretical maximum average bandwidth is $80Mb/s$. Figure 5 illustrates the relation between $\mathcal{P}_n(b)$ and the number of circuits. It can be observed that the probability $\mathcal{P}_n(16)$ approaches 100% when a malicious Tor client creates around 260 circuits, while the probability $\mathcal{P}_n(80)$ approaches 100% at the Tor exit router with bandwidth $80Mb/s$ after creating around 45 circuits. Consequently, if we have more bandwidth, we can collect malicious traffic more efficiently.

Now we discuss how long it takes to observe all m types of malicious traffic in Tor. Assume when $n = k$, $\mathcal{P}_{n=k}(b)$ approaches 100%. Therefore, the average time T needed for observing all m types of malicious traffic can be estimated as follows

$$T = max\{E(\mathcal{T}_k^1), \ldots, E(\mathcal{T}_k^i), \ldots, E(\mathcal{T}_k^m)\}, \tag{3}$$

where $E(\mathcal{T}_k^i)$ is the average time for creating k circuits by the i^{th} type of malicious traffic.

Assume the process of creating circuits by malicious traffic is a poisson process in the Tor network. λ_i is the average rate of the i^{th} type of malicious traffic creating circuits. Denote T_1^i as the time required to create the first circuit and T_j^i as the time required creating the j^{th} circuit after the $(j-1)^{th}$ is created. Therefore the time needed for creating k circuits by the i^{th} type of malicious traffic is

$$\mathcal{T}_k^i = T_1^i + \cdots + T_k^i. \tag{4}$$

The properties of the Poisson process tell us $T_1^i, \ldots, T_k^i$ are independently and identically distributed and $E(T_j^i) = 1/\lambda_i$. Therefore, we can have

$$E(\mathcal{T}_k^i) = k/\lambda_i. \tag{5}$$

Then the average time T needed for observing all m types of traffic is the largest $E(\mathcal{T}_k^i)$ required for observing different types of malicious traffic,

$$T = max\{\frac{k}{\lambda_1}, \frac{k}{\lambda_2}, \ldots, \frac{k}{\lambda_m}\}. \tag{6}$$

We need to profile malicious traffic in the Tor network in order to obtain T. However Equation (6) tells us that the average time of observing malicious traffic is inversely proportional to its circuit creation rate. Different types of malicious traffic will be observed at a Tor exit router given enough time if the malicious traffic is active.

IV. Statistical Analysis of Malicious Traffic Over Tor

In this section, we first show the statistics of traffic protocols over Tor. We then study the alerts and divide them into several groups. At last, we investigate severe malware activities. Note that given the huge volume of traffic, we only used some examples of major threats (e.g., botnet traffic) to show the malicious traffic going through Tor exits. In addition, although IDS may produce false alarms, we believe that the problem is outweighed by the benefit of using an IDS since more people will be willing to host Tor exit routers without concerns of administrative and legal issues.

TABLE I
Datasets

Dataset	IDS Ruleset	Period	Size (TB)
Dataset 1	ETOpen	Oct. 03, 2012 ∼ Nov. 12, 2012	3.95
Dataset 2	ETPro	Jun. 12, 2013 ∼ Jul. 17, 2013	2.97

TABLE II
Network Statistics (Dataset 2)

Protocol	Size (GB)	Packets(Million)	Flows(Thousand)
Email	3.33(1.36%)	47.69(0.86%)	9765.34(11.35%)
SSL/SSH/VPN	6.94(2.83%)	931.31(16.75%)	2288.76(2.66%)
P2P/File Sharing	180.75(73.74%)	3190.5(57.4%)	44520.2(51.74%)
HTTP	19.65(8.02%)	109.81(1.98%)	9853.1(11.45%)
Well-known	1.75(0.71%)	76.76(1.38%)	286.6(0.33%)
Unknown	32.2(13.14%)	1196.59(21.53%)	18846.14(21.9%)
Total	245.11	5558.47	86048.49

A. Discovered Traffic

We conducted our experiments with *TorWard* during two periods: from October 3, 2012 to November 12, 2012, and from June 12, 2013 to July 17, 2013. The traffic during the two periods is stored in dataset 1 and dataset 2, respectively. Table I describes the two datasets in detail. We applied a free version of IDS ruleset, i.e., Emerging Threats ETOpen [25], to obtain alerts for dataset 1. To discover more malicious traffic, a commercial product of IDS ruleset, i.e., Emerging Threats ETPro [25], was used to obtain alerts for dataset 2. We observed similar traffic patterns in the two datasets, and we will focus on presenting dataset 2 in this paper for brevity.

We apply the deep packet inspection library *nDPI* [30] to dataset 2 to derive the statistics of traffic protocols. As we mentioned in Section III-A, the traffic traversing our Tor exit node consists of inbound and outbound Tor traffic. To identify inbound Tor traffic, we use TShark's protocol filter [31] to analyze original traffic and find that around 50% traffic is TLS (Transport Layer Security) traffic, which is used by Tor to encrypt the inbound Tor traffic. After filtering the Tor TLS traffic, we employ a DPI program and obtain the statistical results as shown in Table II. It can be observed from the table that most of recognized traffic by nDPI is P2P and file sharing traffic, showing that P2P and file sharing traffic consumes more Tor bandwidth in comparison with the observation in [5] and [6]. The P2P traffic is the source of various copyright infringement issues and is the reason why a Tor exit node is bombarded with various Digital Millennium Copyright Act (DMCA) complaints. Before introducing *TorWard*, we deployed a Tor exit node on a university campus. In less than 12 hours, we received a DMCA takedown from Warner Bros. Entertainment Inc. that the exit node downloaded their copyrighted materials.

One new trend of Tor usage not observed in [5] and [6] is the traffic generated by mobile devices. *Orbot* [32] was released in 2008 and is a Tor client on Android mobile devices. *Onion Browser* [33] is a Tor-based web browser implemented for apple mobile devices. We now know that with the growth of mobile devices, Tor users begin to install Tor client on their mobile devices to protect privacy of their daily communications. If a mobile device is compromised,

TABLE III
CLASSIFICATION OF ALERTS (DATASET 1)

Classification	Number of Alerts	Percentage
Policy-violation	7,185,153	88.52%
Trojan-activity	387,198	4.77%
Unclassified	382,733	4.72%
Attempted-recon	49,376	0.61%
Bad-unknown	46,548	0.57%
Not-suspicious	31,462	0.39%
Misc-activity	29,052	0.36%
Web-application-attack	2,520	0.03%
Misc-attack	2,155	0.03%
Shellcode-detect	310	0.004%
Attempted-user	267	0.003%
Attempted-admin	1	0.00001%
Total	8,116,775	

TABLE IV
CLASSIFICATION OF ALERTS (DATASET 2)

Classification	Number of Alerts	Percentage
Policy-violation	2,828,285	78.03%
Trojan-activity	325,969	8.99%
Unclassified	214,510	5.92%
Bad-unknown	115,640	3.19%
Not-suspicious	42,782	1.18%
Attempted-recon	42,487	1.17%
Misc-activity	33,583	0.93%
Misc-attack	12,103	0.33%
Protocol-command-decode	6,372	0.18%
Web-application-attack	2,151	0.06%
Shellcode-detect	682	0.02%
Attempted-user	110	0.003%
Attempted-admin	14	0.0004%
Network-scan	9	0.0002%
Attempted-dos	2	0.00006%
Web-application-activity	1	0.00002%
Total	3,624,700	

malware on the mobile device may now route their traffic through Tor.

B. Alert Classification

TorWard allows us to monitor outbound Tor traffic from our exit router by using Suricata [24], a well known IDS. To study malicious activities, we applied two distinct IDS rulesets, ETOpen and ETPro, to the two datasets and automatically updated the ruleset periodically. In the following, we introduce various classes of raised alerts shown in Tables III and IV.

Unclassified alerts are mainly made up of Russian Business Network (RBN) and Malvertiser. RBN is known for hosting illegal contents, such as child pornography, phishing, spam, and malware [34].

Policy-violation alerts consist of P2P alerts, online games (e.g., Battle.net) alerts, various chat alerts, and others. We found that around 99% of alerts in this category are actually generated by P2P traffic.

Misc-attack alerts are generated for blacklisted hosts. The IDS rules contain IP addresses of hosts or netblocks, which are known to be bots, phishing sites, professional spammers, and so on. The blacklist is obtained from various sources, including Dshield [35], Spamhaus [36], Brute Force Blocker [37], OpenBL.org [38], C.I.Army [39], and the Emerging Threats Sandnet and SidReporter project [25].

Alerts for *trojan-activity* are for various detected malwares. We observed alerts for the Ngrbot channel, IRC channel on a non-standard port, Zeus, various potential IRC bot user names, Ruskill/Palevo download commands, iebar spyware, hotbar spyware, simbar spyware, Zango Seekmo bar spyware, fun web products spyware, AskSearch toolbar spyware, Cycbot/Bifrose/Kryptic traffic, Vobfus/Changeup/Chinky download commands, known hostile domain ilo.brenz.pl lookup, DNS queries for .su (Soviet Union) that is considered as related to Malware, and many others. We categorized these malwares into several groups in Table V. Due to the space limit, we do not list the full table of the classification, which is available upon request. Because Tor clients are now available for mobile devices, we discovered several well-known Android malwares as well.

Misc-activity comprises various IRC commands, packed executable downloads, .cn and .ru malware related domains, .dyndns.org DNS lookup, and potential port scan behavior on remote port 135, 139, 445, and 1433.

Bad-unknown consists of diverse DNS queries and HTTP requests for suspicious domains, such as .co.cc, .tk, .org.pl, .cz.cc, .co.tv, .xe.cx, and others. These suspicious domains can be used by C&C servers. We also find alerts from HTTP redirection to Sutra TDS (Traffic Direction System) that might force a client to download malware.

Shellcode-detect alerts indicate that the content of the traffic contains various no operation (NOOP) strings. The attacker can send long strings of NOOPs to overflow the buffer and gain root access to an x86 Linux system. We also find heap spray string related alerts.

Not-suspicious alerts are for IP addresses blacklisted by Abuseat.org, Robtex.com and Sorbs.net for spam emails. Because Tor exit routers may relay spam emails, their IP addresses are also blacklisted and recommended for blocking by those websites.

Attempted-recon alerts include the potential SSH port scans. The alerts suggest that some Tor clients probably attempt to scan the SSH port. Also, we find the activities of retrieving the external IP addresses of the Tor exit router from web sites such as showip.net, myip.dnsomatic.com, cmyip.com, ipchicken.com, whatismyip.com, showmyip.com, and others. Because a number of malwares try to get the external IP address once the victim host is infected, the inquiry traffic might be rerouted into the Tor network and relayed by our exit Tor router.

Alerts for *attempted-admin* include those for a type of buffer overflow vulnerability caused by a boundary error in the GIF image processing of Netscape extension 2. We also discovered http post requests with negative content length that can cause buffer overflow at a web server. Microsoft DirectShow AVI file buffer overflow alerts were found. This vulnerability allows a remote attacker to execute malicious code at a Tor client.

Web-application-attack alerts are for two types of attacks: attacks from the client side and attacks from the server side. We observed that the alerts were from the client side, including SQL injection attacks by using the Havij SQL injection tool.

TABLE V
MALWARE DISCOVERED THROUGH ALERTS

Platform	Classification	Example Malware	Number of Malware
PC	Virus	Win32/Virut.A, Win32.Sality-GR, Win32/Sality.AM, W32/Virut.n.gen, VirTool.Win32/VBInject.gen!DM, Brontok, Luder.B	7
	Worm	Win32.Duptwux/Ganelp, Win32/Fujacks, Win32/Ruskill/Palevo, Win32/Gamarue.F, Win32.AutoTsifiri.n, Win32/Cridex.E, Worm.Win32.Balucaf.A, Koobface Beaconing (getexe), Vobfus/Changeup/Chinky, Win32/Zhelatin, Possible Bobax	11
	Trojan	Win32/Cutwail.BE, Zbot (AS9121), Win32/Tibs, Win32.Fareit.A/Pony, Win32/Sinowal/sinonet/mebroot/Torpig, etc.	88
	Backdoor	Win32/Prosti, Win32/Hupigon.CK, Win32/Bifrose/Cycbot, Win32.Aldibot.A, Win32.Gh0st, Win32/Kbot, etc.	39
	Spyware	Baidu.com Spyware Bar, AskSearch Toolbar Spyware User-Agent, Casalemedia Spyware, Alexa Search Toolbar User-Agent (Alexa Toolbar), ISearchTech.com XXXPorn-Toolbar Activity (MyApp), etc.	45
	Bot	Yoyo-DDoS Bot, JKDDOS DDoS Bot, BlackEnergy DDoS Bot, Illusion Bot, Zeus Bot, P2P Zeus, Darkness DDoS Bot, SpyEye, IMDDOS Botnet User-Agent STORMDDOS, Dropper.Win32.Agent.bpxo, Win32/Dorkbot(NgrBot), Andromeda, Known Skunkx DDOS Bot User-Agent Cyberdog, MRSPUTNIK, ZeroAccess/Sirefef/MAX++/Jorik/Smadow	14
	Adware	W32/OpenCandy, Adware.Gen5, Adware.iBryte.B, Win32.AdWare.iBryte.C, ADWARE/InstallCore.Gen, Win32/InstallMonetizer.AC, Adware.Solimba, AdWare.Win32.Eorezo, BInet Information Install, Adware/Win32.MediaGet User-Agent (mediaget), Common Adware Library ISX User Agent, W32/GameVance Adware	12
Mobile Device	Malware	Android/Qdplugin.A, Android/Adware.AirPush.D, Android.Troj.FakeSms.a, Android/Plankton.P, Android.Plankton/Tonclank	5

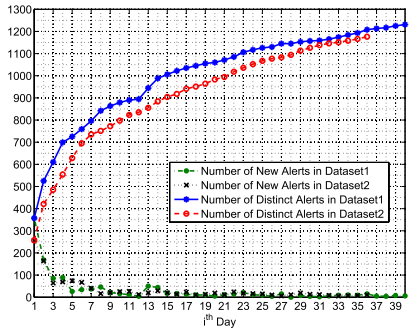

Fig. 6. The relation between number of discovered alerts and $i^{th} day$.

The alerts from the server side were from malware in the web page and cross-site scripting attacks, which allow the malicious code to be executed by a Tor client browser.

Attempted-user alerts are triggered by the inbound traffic, which attempts to use various vulnerabilities of web browsers (e.g., Mazilla Firefox and Microsoft Internet Explorer) at the Tor client side to launch attacks. The remote attacker may take advantage of these vulnerabilities to execute the malicious code and control the machine where the Tor client is hosted. For example, we found the alerts that report a remote server's attempt to return a file embedded with a Class ID (CLSID) to the web client at the Tor client side. There are also alerts related to cross-site scripting (CSS) attacks.

Attempted-dos alerts show that malicious code is detected in the incoming traffic, exploiting the stack exhaustion vulnerability in the Microsoft Internet Explorer Script Engine. If the Tor client uses a vulnerable version of the web client to open the malicious web page, the web client can be terminated as a result of DoS attacks.

C. Malicious Traffic Statistics

In Figure 6, the two upward curves show the cumulative number of distinct alerts from datasets 1 and 2, respectively. They increase very slowly after several days. The two downward curves show the number of daily discovered new alerts. Few new alerts are observed after a few days. These results match our theoretical analysis in Section III-C. In a first few days, we have captured most of alerts over Tor. Apparently, new malicious traffic have been emerging according to Figure 6.

According to the IDS ruleset classification [40], we categorized the discovered alerts into several categories. Tables III and IV list the number of alerts of various groups collected in two datasets, and detailed description is shown below. It is worth noting that we applied two distinct IDS rulesets for these two datasets. In Tables III and IV, there are $8,116,775$ alerts for dataset 1 and $3,624,700$ alerts for dataset 2. Policy-violation alerts have the largest percentage, and they are incurred by P2P traffic. Alerts related to malware include *unclassified*, *misc-attack*, *trojan-activity*, *not-suspicious*, and *misc-activity*. These alerts involve well-known or potential IP addresses of C&C servers, well-known malicious traffic, suspicious DNS query traffic, spam traffic, and suspicious IRC traffic.

To understand the traffic volume in different categories, we compute the volume of incoming and outgoing traffic from raw data based on the alerts. Tables VI and VII give the traffic information of dataset 2. The results are sorted based on the priority of the ruleset [40]. From Tables VI and VII, we can observe that around $(2.5GB + 162GB) = 164.5GB$ out of $2.97TB$ traffic, i.e., around 5.5% traffic in dataset 2, can trigger the alerts, while around $(16GB + 375GB) = 391GB$ out of $3.95TB$ traffic, i.e., around 10% traffic in dataset 1, can trigger the alerts [3]. In addition, the policy-violation traffic is the most dominant one, which was mainly caused by P2P traffic. Then, the second dominant traffic is trojan-activity traffic. In addition, the volume of traffic generating high priority alerts is much larger than the volume of other traffic.

TABLE VI
INCOMING TRAFFIC STATISTICS (DATASET 2)

Classification	Size (Bytes)	Percentage	Priority
Shellcode-detect	238,734,330	9.55%	High
Trojan-activity	220,219,632	8.81%	High
Policy-violation	66,546,599	2.66%	High
Web-application-attack	16,888,851	0.68%	High
Attempted-user	70,334,933	2.81%	High
Attempted-admin	48,477,268	1.94%	High
Misc-attack	552,748,810	21.40%	Medium
Bad-unknown	200,336,881	8.02%	Medium
Web-application-activity	595	0.00002%	Medium
Attempted-recon	178	0.000007%	Medium
Misc-activity	534,901,914	22.12%	Low
Not-suspicious	30,105,913	1.20%	Low
Protocol-command-decode	8,124,809	0.33%	Low
Network-scan	1,253	0.00005%	Low
Unclassified	511,917,854	20.48%	Unknown
Total	2,499,339,820		

TABLE VII
OUTGOING TRAFFIC STATISTICS (DATASET 2)

Classification	Size (Bytes)	Percentage	Priority
Policy-violation	159,692,890,115	98.52%	High
Trojan-activity	2,004,499,807	1.24%	High
Attempted-user	39,242	0.00002%	High
Web-application-attack	7,631	0.000005%	High
Bad-unknown	174,978,188	0.11%	Medium
Attempted-recon	35,503,428	0.02%	Medium
Attempted-dos	5,373	0.000003%	Medium
Not-suspicious	166,439,974	0.10%	Low
Misc-activity	10,796,084	0.007%	Low
Protocol-command-decode	4,586,076	0.003%	Low
Total	162,089,745,918		

Based on the diverse alerts, we conclude that outbound Tor traffic consists of numerous malicious traffic, which may potentially incriminate the party who hosts a Tor exit router. In addition, third-party plug-ins of various browsers used by some Tor users may leak their private information. In the following subsection, we further explore the issues incurred by malware activities to reveal the impact of the malicious traffic.

D. Malware Activities

As shown in Table V, we discovered various activities associated with malwares from the reported alerts, including the communication between malware and C&C server, DoS attacks, Spams, and others.

Communication Between Malware and Command and Control Server: Some malwares are designed to connect to a C&C server in order to report the information retrieved from the victim machine, update the malware, download the configuration file, and perform other operations. To hide the communication between malware and the C&C server, malware authors may adopt Tor to hide malicious traffic and protect the real location of the C&C server from being discovered. If the malware chooses our Tor exit router, the malicious traffic will traverse the Tor circuit and establish the connection to the C&C server through our exit Tor router. Hence, our exit Tor router can detect such malicious traffic. In dataset 2, we discovered 622 C&C server IP addresses based on check-in messages from more than 70 different

```
GET /outlawz/mainp/gate.php?guid=GT!GT-
FDCCD9A7405D!30457F77&ver=10120&stat=ONLINE&ie=6.0.2900.5512&os=5.1.2600&ut=A
dmin&cpu=61&ccrc=8115AE02&md5=16ab5c0e831612b94e193282537b97e8 HTTP/1.1
Host: outlawyoung972.mobi
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
Connection: Close
```

Fig. 7. Spyeye checkin.

```
PASS ngrBot
:how.dare.you NOTICE AUTH :*** Looking up your hostname...
NICK {USA|W7u}ayhqcku
USER ayhqcku 0 0 :ayhqcku
:how.dare.you NOTICE AUTH :*** Couldn't resolve your hostname; using your IP address instead
:how.dare.you 001 {USA|W7u}ayhqcku
:how.dare.you 002 {USA|W7u}ayhqcku :              M0dded by uNkn0wn Crew
:how.dare.you 003 {USA|W7u}ayhqcku
:how.dare.you 004 {USA|W7u}ayhqcku :        www.uNkn0wn.eu - iD@uNkn0wn.eu
:how.dare.you 005 {USA|W7u}ayhqcku
:how.dare.you 005 {USA|W7u}ayhqcku
:how.dare.you 005 {USA|W7u}ayhqcku
:how.dare.you 422 {USA|W7u}ayhqcku :MOTD File is missing
:{USA|W7u}ayhqcku MODE {USA|W7u}ayhqcku :+iwG
JOIN ##Redrm-002## redem
JOIN ##Redrm-002## redem
:{USA|W7u}ayhqcku!ayhqcku@61.32.75.88 JOIN :##Redrm-002##
:how.dare.you 332 {USA|W7u}ayhqcku ##Redrm-002## :!NAZEL http://hotfile.com/dl/146666161/
add093d/FACEBOOK-DSC009854162487312.jpg.exe
:how.dare.you 333 {USA|W7u}ayhqcku ##Redrm-002## xXx 1329485141
```

Fig. 8. Ngrbot.

known malware, 59 different IP addresses of known compromised or hostile hosts that might be deployed as a C&C server, 71 C&C Server IP addresses reported by Shadowserver [41], and 93 IP addresses obtained from various well-known trackers (e.g., Zeus, Spyeye, and Palevo trackers) that report C&C servers' IP addresses.

We now show a few examples found in our datasets on how malwares communicate with their C&C servers. In Figure 7, a Spyeye bot is connecting to its C&C server to report the information of the victim machine. According to the format of SpyEye C&C Message [42], we can parse the information that includes a unique identifier (guid=GT!GT-FDCCD9A7405D!30457F77), the version of the bot infector (ver=10120), the status of the bot (stat=ONLINE), the version of Internet Explorer (ie=6.0.2900.5512), the version of Microsoft Windows operating system (os=5.1.2600), the type of the current user on the victim machine (ut=Admin), the CPU load (cpu=61), the CRC32 taken from the last four bytes of the bot configuration file (ccrc=8115AE02), and the md5 of the bot infector (md5=16ab5c0e831612b94e193282537b97e8). Figure 8 shows that a Ngrbot logs into a IRC server, joins a chat room and then receives a command to download another malware. We found malicious traffic from mobile devices as well. As an example, Figure 9 illustrates the malware communicating with the remote server by using HTTP protocol.

DoS Attacks: A bot master can control a large number of bots and malware to perform a DoS attack through Tor. For example, in our measurements, we discovered 72,894 DoS attack alerts of Yoyo-DDoS bot where 457 distinct destinations are found. Yoyo-DDos bots can receive the command of attacking a target server from the bot master and then continuously send HTTP requests to the target server so as to launch HTTP flood attacks. The target servers of 96% DDoS attacks that we found are located in two countries, the Unite States and China.

Spam Traffic: We found 40,834 related spam alerts and 8,186 distinct email server IP addresses from 115 different countries in dataset 2. As we can see from Table VIII, 89.02% alerts originate from only 10 countries, while around 50%

```
POST /ProtocolGW/protocol/eulastatus HTTP/1.1
device-id: oaA2oxbJ574JFTt0bvnHmGJyhC8%3D
protocol-version: 2.0.1
User-Agent: Mozilla/5.0 (Linux; U; Android 4.1.2; en-gb; GT-I9105 Build/JZO54K)
AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobile Safari/534.30
Content-Type: application/json
Accept-Encoding: gzip
Accept: application/json
Content-Length: 579
Content-Encoding: gzip
Host: www.apperhand.com
Connection: close

Send DATA.


HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Type: application/json
Transfer-Encoding: chunked
Date: Sat, 22 Jun 2013 08:45:49 GMT
Connection: close

Response DATA.
```

Fig. 9. Mobile Malware (Android/Plankton.P).

TABLE VIII
SPAM ALERT STATISTICS (DATASET 2)

Country	Number of Alerts	Number of Distinct IP Addresses
Japan	20,701 (50.70%)	1,377 (16.82%)
China	5,174 (12.67%)	759 (9.27%)
United States	4,079 (9.99%)	1,782 (21.77%)
Korea, Republic of	1,847 (4.52%)	228 (2.79%)
Russian	1,308 (3.20%)	223 (2.72%)
Canada	1,216 (2.98%)	231 (2.82%)
United Kingdom	634 (1.55%)	281 (3.43%)
Germany	621 (1.52%)	417 (5.09%)
Philippines	407 (1.00%)	381 (4.65%)
France	376 (0.92%)	181 (2.21%)
Others	4,471 (10.95%)	2,326 (28.41%)
Total	40,834	8,186

email servers are from only three countries. Due to the large number of spams from Tor network, many email servers deny the email relayed from the Tor network. This hurts benign Tor users who send email through Tor.

Bitcoin Pool Traffic: We discovered 11,216 alerts related to communication between bitcoin miner and distinct bitcoin pools in dataset 2. Bitcoin is a decentralized electronic currency. To generate new bitcoins, a node should solve a mathematical problem, i.e., creating a new block to show a proof of work. According to [43], a new block yields around 25 bitcoins, which is about $25 * 96 = 2400$ US dollars in terms of current price in the bitcoin exchange market. Nonetheless, it is difficult for a computer with limited computation power to generate a block. To address this issue, a bitcoin pool server is used to split a block into pieces of small work and let multiple users to work together to mine bitcoins. Hence, some malicious botnets exploit the computational power of victim machines to make profit by mining bitcoin. For example, Skynet bots [11], [12] can deploy bitcoin miner in the victim machines. Hence, the alerts from our datasets suggest that some victim machines are installed with a bitcoin miner and communicate with a bitcoin pool server.

E. Botnet Over Tor

Our experiments disclose that various malicious traffic (e.g., P2P, botnet and spam) are routed through Tor.

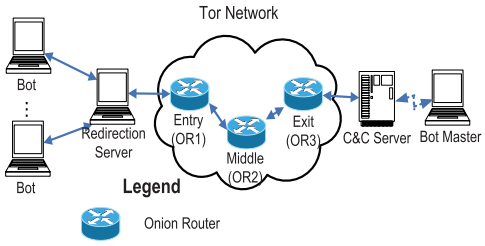

Fig. 10. Botnet over Tor (scene 1).

Our experimental results, further detailed in later sections, suggest that a botnet owner may use the Tor network to hide the communication between bots, botmaster and the C&C server. Before we introduce mechanisms to trace back botnet traffic over Tor, we discuss possible strategies that botnets may use to abuse Tor as anonymous stepping stones to hide the botmaster and the C&C server.

A botnet can use the Tor network and hide communication in two approaches. First, bots are installed with the Tor client software. By setting the firewall rules, a bot can work as a transparent proxy to force its traffic to go through Tor. Second, a bot can be configured to connect to a traffic redirection server, which forwards the bot traffic into the Tor network to reach the real C&C server. We have managed to deploy such a traffic redirection server to forward the traffic between bots and Tor network. We integrate a reverse proxy Pen [44], Tor client and transparent socks proxying library (tsocks [45]) with a traffic redirection server. The forwarding destination of the reverse proxy is the C&C server. The reverse proxy is configured to transparently forward the bot traffic to the socks proxy of the Tor client through tsocks. We deploy UnrealIRCd [46] as a remote IRC C&C server and install Ngrbot in a virtual machine. We set the C&C server option of the malware as the redirection server. The bot first connects to the redirection server and is then redirected to the hidden IRC C&C server through Tor network. Ultimately, the bot can obtain the commands from the botmaster through the C&C server.

The first approach of bundling bots with Tor and transparent proxy is harder to deploy. In particular, while most reported bots run over Windows, there is no such transparent proxy for Windows systems. The attacker may modify the code to embed the proxy functionality into botware. Nonetheless, botware source code may not be always available. A builder of botware often does not have the option of using Tor.

The second approach of using a redirection server to relay the bot traffic to the C&C server is more realistic due to the ease of deploying the redirection server in diverse operating systems. Figure 10 illustrates this deployment. With this method, the attacker can hide the real C&C server even if bots are discovered. The above mentioned botnet over Tor deployed by us is a simplified version of the Zeus botnet using a hidden server over Tor [8], in which an attacker deploys the C&C server of Zeus as a Tor hidden server, and bots communicate with the hidden C&C server through Tor2Web [9]. Tor2Web is a third-party tool designed to help access the hidden web servers without a Tor client and facilitate the Zeus bots, which do not support the proxy functionality to

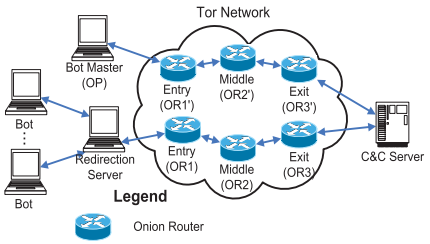

Fig. 11. Botnet over Tor (scene 2).

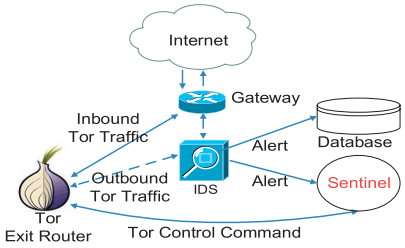

Fig. 12. Blocking malicious traffic.

connect to the hidden C&C server. Our traffic redirection server works as a private Tor2Web to forward the traffic to a specific destination. To resist the single point of failure (SPOF), the botnet owner can deploy several backup redirection servers. For example, the builder of Ngrbot provides three backup C&C server options.

A botmaster can also connect to the C&C server through Tor and hides itself from traceback, as illustrated in Figure 11. In our experiments, we found all botnet traffic passing through our Tor exit router is not encrypted. Hence, it is possible to detect the botmaster traffic using IDS. For example, in a centralized IRC botnet, the botmaster actively sends commands to bots using IRC *PRIVMSG* messages through the C&C channel, while bots wait for the commands from the botmaster. We can configure IDS to monitor the incoming IRC *PRIVMSG* messages at the Tor exit router and detect the potential bot commands.

V. Blocking and Tracing Malicious Traffic Over Tor

In this section, we first introduce *TorWard* as a system for blocking malicious traffic detected by IDS at Tor exit routers. Blocking is a passive countermeasure to various malicious activities. To further deter illegal activities and promote legal use of Tor, *TorWard* can also be used to trace severe attacks. This is possible and legal if exit and entry routers collaborate. When the suspect IP is identified, we can refer the case to law enforcement for further collection of evidence and even prosecution.

A. Blocking Malicious Traffic

TorWard can be used to block potential malicious traffic at Tor exit routers. We use the intrusion detection alerts from IDS and make a decision of either disconnecting or keeping the corresponding outbound traffic through our custom Tor control protocol. Figure 12 illustrates the structure of *TorWard* for this purpose.

In *TorWard* defense system, there are four components: a Tor exit router, an IDS, a sentinel, and a database. The IDS monitors traffic passing through the exit router and sends alerts to the sentinel for real-time processing and to the database for off-line analysis. The sentinel retrieves the destination of a suspect connection from the alerts and sends our customized disconnection command to the Tor exit router through the Tor control protocol. The Tor exit router obtains the IP address and port, and then searches its connection list. Once the suspect connection is found, the corresponding connection will be terminated.

For IDS, we can choose either Suricata [24] or Snort. The IDS can be configured to send the alerts through a Unix domain socket to the sentinel. Snort has a configuration option of using Unix domain socket. Suricata does not have the Unix domain socket functionality. Nonetheless, we can configure Suricata to record the alerts in a binary file. Barnyard2 [26] can then be used to parse the file and send the alerts to the sentinel with the "alert_unixsock" option in its configuration file. Both Snort and Suricata are signature-based IDS and we adopt the Vulnerability Research Team (VRT) rules [47] and Emerging Threats (ET) rules [25] for them. Rules can also be customized in the IDS configuration file to detect certain category of threats. For example, if we concern about the P2P traffic through the exit Tor router, we can include only relevant P2P rules into the IDS configuration file. It is worth noting that our developed system is generic and other anomaly-based detection algorithms can be deployed.

Upon obtaining alerts from IDS, sentinel processes the alerts, retrieves the IP address and the port number of the potential malicious traffic, and disconnects the corresponding connection through our custom Tor control protocol. We use the Tor control protocol [48] to send our customized commands to the exit router, which executes the designated task. For example, command "CLOSEEXITCONN ip port" is added to tear down the suspicious connection. Once the exit router receives this command, it sends the remote client a "RELAY_COMMAND_END" cell with the reason "REASON_CONNECTREFUSED" to inform that the connection is closed.

To carry out an off-line analysis of alerts, the IDS is configured to record the alerts in a database. In our case, MySQL is used. The IDS stores alerts in a binary file, and Barnyard2 reads the file and sends the alerts to the database. The database is also managed by a front application, Basic Analysis and Security Engine (BASE) [27]. It is worth noting that the database is not necessarily an essential component for *TorWard*, which can work smoothly without the database.

TorWard cannot prevent the malicious traffic from the source, although it can effectively disrupt the malicious traffic at the exit router. According to our observation, the involved circuit is not really destroyed by *TorWard*. Instead, the remote Tor client may choose a new Tor router, which is not equipped with *TorWard*, and adds it into this current circuit as the new exit router. Then, the remote Tor client will use the four-hop circuit to communicate with the destination again. To effectively block malicious traffic, we should track down the offending Tor client. In the next subsection, we design

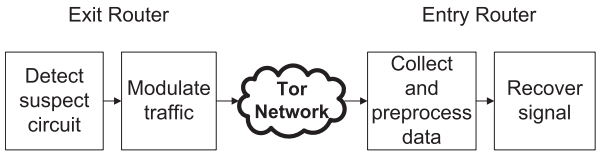

Fig. 13. Workflow of the DTMF signaling based approach.

an approach and take the IRC botnet traffic as an example to show how to discover the remote botnet hosts.

B. Dual-Tone Multi-Frequency Signaling Based Traceback

The goal of traceback is to correlate botnet traffic at an exit router and that at an entry router across Tor. For this purpose, we adopt the dual-tone multi-frequency (DTMF) signaling approach [49], which has been used for telecommunication signaling over analog telephone lines in the voice-frequency band between telephone handsets, other communications devices and the switching center. In DTMF, to send a single key such as "9", we select a low frequency and a high frequency, and send a sinusoidal tone of the two frequencies. The tone is decoded by the switching center to determine the key that was transmitted. The original DTMF adopts 8 frequencies to represent $4 \times 4 = 16$ keys. Inspired by DTMF, we design a DTMF signaling based approach to tracing the botmaster or bots over Tor. In the following, we will introduce the basic idea and the workflow, and discuss the critical issues of using DTMF for traffic traceback.

1) Basic Idea: Recall that in the cyber attack scenario shown in Figure 11, the exit router discovers botnet traffic from a suspect circuit. We want to find the offending entity that generates the bot traffic at the other side of Tor, which can be a redirection server or botmaster. Our goal is to find the botnet IP address through our signaling approach. Assume that we control a small percentage of exit and entry onion routers by donating computers to Tor. Because Tor is operated in a voluntary manner, this assumption is valid in practice and has been widely used [15], [50]–[52]. Notice that the traceback helps the law enforcement possess the capability of identifying the malicious source in case of severe crime activties such as child pornography downloading observed at an exit router.

The basic idea of tracing botnet traffic is that at the controlled exit router, we first inject extra cells at alternating frequencies that represents a signal into the suspect circuit and then attempt to confirm the signal at our controlled entry routers. If one entry detects the signal, this entry will be used to identify the IP addresses of botnet hosts, which actually creat the suspect circuit. Our DTMF signaling uses two different frequencies to represent bit 0 and bit 1, respectively. A signal is a sequence of binary bits.

2) Workflow of DTMF Signaling: Figure 13 illustrates the workflow of the DTMF signaling method. We now introduce individual steps in detail.

Step 1 (Detecting Suspect Circuit): With the help of an IDS, the exit router can find the suspect connection and corresponding circuit, which transmits the IRC bot traffic. The IP address and port are obtained accordingly.

Step 2 (Modulating Traffic): Once a suspect circuit is detected, we can inject artificial cells into the circuit and start the traceback procedure. For an IRC channel, messages in the injected cells should not be displayed. An empty IRC *PRIVMSG* message works for this purpose. Two distinct frequencies for transmitting cells, denoted as **feature frequencies**, are selected to represent 0 and 1, respectively, and modulate a signal into the injected traffic. For example, to encode a bit 0, we send cells with an interval time of $500ms$ (i.e., a feature frequency of 2Hz); to encode bit 1, we send cells with an interval time of $333.33ms$ (i.e., a feature frequency of 3Hz). Each bit can last for a longer interval, such as 2 seconds, denoted as *bit interval*.

Step 3 (Collecting and Pre-Processing Data): At our controlled entry routers, we record cells for each circuit in order to derive the feature frequency embedded in cells and recover a signal. A traffic volume time series is derived by counting the number of cells in a sampling interval T_s, corresponding to the sampling frequency $F_s = 1/T_s$. We denote the time series as

$$X(F_0, F_1, F_s) = \{x_1, \ldots, x_N\}, \tag{7}$$

where F_0 and F_1 are the two feature frequencies to represent bits 0 and 1, N is the sample size, and x_i is the number of cells in the i^{th} sampling interval.

Sampling frequency F_s has to be carefully selected to recover an embedded feature frequency. We expect that the feature frequency should show a strong amplitude than noise around the expected feature frequency in the frequency domain. In order to recover the feature frequency F_I (I is 0 or 1), sampling frequency F_s (corresponding to the sampling interval T_s) must be carefully chosen. According to Nyquist sampling theory, we have

$$F_s \geq 2F_I. \tag{8}$$

One issue of recovering feature frequencies is how to synchronize modulation and demodulation at the exit and entry. That is, we want to know the time when (at which cell) cells for 0 and 1 start. This issue can be solved in our case because we control the cell sending at the exit router. Botnet traffic into the suspect circuit toward the entry can be blocked before we inject the traceback cells into the circuit. Hence, an appropriate silence period such as 1 second can be introduced to indicate the starting time of transmitting the signal. Recall that the entry must monitor all its circuits. Circuits without such a silence period will be eliminated immediately. For circuits with the silence period, we can count the cells. Because we know the time when "1" and "0" start and end, respectively, we can carry out the Fourier Transform on the corresponding traffic segment.

Step 4 (Recovering Signal): In this step, we apply the Fourier transform to $X(F_0, F_1, F_s)$. Recall that we introduce periodicity while sending cells at the entry. If a circuit indeed carries the botnet traffic, strong amplitudes will be observed at feature frequencies F_0 and F_1, which correspond to bits 0 and 1, respectively. The IP address that creates the suspect circuit will disclose the suspect botnet.

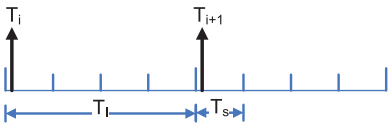

Fig. 14. Sampling cells in noise-free environment.

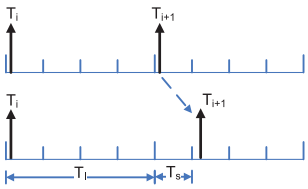

Fig. 15. Shifted cells in noisy environment.

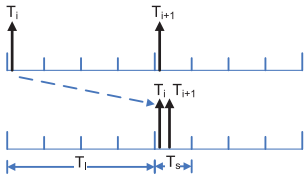

Fig. 16. Merged period in noisy environment.

C. Issues of DTMF for Traceback

Several critical issues listed as follows should be addressed when using DTMF for traceback.

- We modulate a signal bit into cells by sending cells at a specific feature frequency. When the cells pass through Tor, how does network dynamics affect cell timing? If the cell timing is changed, the feature frequency could be distorted.
- In our traceback strategy, feature frequencies will be identified in the frequency domain. Recall that we may have to monitor multiple circuits and identify the circuit that carries the feature frequencies. What is the rule for deciding whether a feature frequency exists given that the network dynamics may have distorted cell timings along the circuit?
- How do we choose feature frequencies? Are those frequencies are arbitrarily selected?

We will answer these questions in the next section.

VI. Analysis

In this section, we address the issues of using the DTMF based approach to trace botnet traffic: (i) the impact of noise, (ii) the decision rule for recognizing a signal bit, and (iii) the selection of feature frequencies. Also, we will investigate two performance metrics: detection rate and false positive rate.

A. Interference of Noise

When injected cells into a suspect circuit pass through Tor, the cells can be interfered with in various ways. Figure 14 illustrates the case in a noise-free environment. Two consecutive cells are observed in the correct sampling interval T_s. By using the Fourier transform, we can derive the correct feature frequency. Nonetheless, the Tor router may be congested and the Internet may also delay the cells (wrapped in network packets) randomly, the inter-arrival times of the cells will be disturbed. The feature frequency will be affected accordingly.

To better understand the impact of noise on feature frequencies, we study the following two cases:

Case 1 (Cell Shifting): Denote the time instants when cells arrive at OR1 as $\{T_1, \ldots, T_m\}$, where m is the total number of cells. Denote T_α as the one-way trip delay between OR3 and OR2, and T_β as the one-way trip delay between OR2 and OR1. Denote the processing time of data at OR2 as T_η. Hence, the relationship between T_i and T_{i+1} can be represented by,

$$T_{i+1} = T_i + T_I + T_\alpha + T_\beta + T_\eta, \tag{9}$$

where $1 \leq i < m$ and T_I is the interval time for sending cells at a feature frequency. Because the delay introduced by network dynamics and the load of middle onion router is uncertain, we denote the uncertain factor as random variable T_θ, where $T_\theta = T_\alpha + T_\beta + T_\eta$. Then, we have

$$T_{i+1} = T_i + T_I + T_\theta. \tag{10}$$

Even with noise, each cell can be in the correct sampling interval if $T_i + T_I < T_{i+1} < T_i + T_I + T_s$, as illustrated in Figure 14. If the interference is large enough to cause $T_\theta > T_s$, then the $(i+1)^{th}$ cell will be shifted into the next segment, as illustrated in Figure 15, or even further. In this case, the amplitude of the feature frequency is reduced or the frequency itself can be changed.

Case 2 (Cell Merging at the Middle Onion Router): Denote the time instants when cells arrive at OR2 as $\{T'_1, \ldots, T'_m\}$, where m is the number of the cells. Denote the interval-arrival time between T'_i and T'_{i+1} as T_ρ, where $T_\rho = T_I + T_\alpha$. Then, the relationship between T'_i and T'_{i+1} can be represented by,

$$T'_{i+1} = T'_i + T_I + T_\alpha. \tag{11}$$

Basically, if the current cell arriving at the middle router can be promptly transmitted to the entry router before the following cell arrives at the middle router, these two cells will not be combined, that is, $T'_i + T_\eta < T'_{i+1}$. Based on Equation (11), we can obtain $T'_i + T_\eta < T'_i + T_I + T_\alpha$ and eventually derive $T_\eta < T_\rho$. Hence, if $T_\eta > T_\rho$, two consecutive cells will be combined, as illustrated in Figure 16. In this case, feature frequency in these two segments will be changed due to the interference of the period.

B. Decision Rule for Recovering the Signal

We now discuss how feature frequencies and the signal are recovered from the traffic volume time series. As we have shown, a feature frequency is introduced by injecting cells periodically. If we treat the traffic volume as a continuous function of time $f(t)$, the Fourier transform of a periodic function $f(t)$ can be represented by a Fourier series as follows,

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{i2\pi \frac{k}{T_I} t} = \sum_{k=-\infty}^{\infty} c_k e^{i2\pi k F_I t}, \tag{12}$$

where T_I is the periodical time for transmitting cells, F_I is the a feature frequency, $e^{ik2\pi F_I t} = \cos(2\pi k F_I t) + i \sin(2\pi k F_I t)$ and c_k is the k^{th} coefficient. Note that *without noise, in the frequency domain, only frequency components at kF_I have no-zero amplitude.* That is, the power P_s of the signal can be written as follows,

$$P_s = \sum_{k=-\infty}^{\infty} |c_k|^2, \tag{13}$$

where $|c_k|^2$ corresponds to the power of the corresponding frequency component.

Nonetheless, network dynamics may distort feature frequencies along the circuit, and noise is added into the power spectrum of the traffic volume function $f'(t)$,

$$f'(t) = f(t) + \zeta, \tag{14}$$

where $f(t)$ is the traffic volume function that we introduce at the entry router. We assume that ζ is *Gaussian white noise* (WGN) with distribution $N(0, \sigma^2)$.

According to Parseval's theorem, we can derive the power P_s' corresponding to $f'(t)$,

$$P_s' = \frac{1}{2\ell} \int_{-\ell}^{\ell} (f(t) + \zeta)^2 \, dt, \tag{15}$$

and the expectation of the signal power is derived by,

$$E(P_s') = E\left(\frac{1}{2\ell} \int_{-\ell}^{\ell} (f(t) + \zeta)^2 \, dt\right) \tag{16}$$

$$= \frac{1}{2\ell} E\left(\int_{-\ell}^{\ell} \left(f(t)^2 + 2f(t)\zeta + \zeta^2\right) dt\right) \tag{17}$$

$$= \frac{1}{2\ell} \int_{-\ell}^{\ell} \left[f(t)^2 + 2E(\zeta) f(t) + E(\zeta^2)\right] dt. \tag{18}$$

Since $E(\zeta) = 0$ and $E(\zeta^2) = \sigma^2$, we have

$$E(P_s') = \frac{1}{2\ell} \left(\int_{-\ell}^{\ell} f(t)^2 dt\right) + \sigma^2, \tag{19}$$

$$= P_s + \sigma^2. \tag{20}$$

Therefore, we can derive the signal-to-noise ratio (SNR),

$$SNR = \frac{E(P_s')}{\sigma^2}. \tag{21}$$

Substituting (20) and (13) into (21), we have

$$SNR = \sum_{k=-\infty}^{\infty} \frac{|c_k|^2}{\sigma^2} + 1. \tag{22}$$

From Equation (22), SNR at each frequency component must be large enough so that the feature frequency can be recognized.

After we apply the Fourier Transform to the suspect traffic volume time series, the feature frequency is expected with large amplitude. We can use a threshold λ to determine whether the feature frequency has a large enough amplitude. That is, if

$$\text{SNR at the expected feature frequency} > \lambda, \tag{23}$$

the feature frequency and corresponding signal bit are recovered. The threshold value can be selected through off-line training.

C. Selection of Feature Frequencies

Because cells may shift or merge along a Tor circuit due to network dynamics, feature frequencies have to be carefully selected to avoid large signal distortion. If a high frequency is used to transmit cells, the interval between cells would be small and cells would be likely to merge at the middle router. Hence, a low frequency is more appropriate for a feature frequency. Nonetheless, a lower frequency implies a longer traceback time.

Because we use two frequencies to represent signal bits 0 and 1, respectively, and recognize them in the frequency domain, the two frequencies must not overlap in the frequency domain. Assume feature frequency $F_0 < F_1$. If $F_1 = kF_0$, where k is a positive integer, according to Equation (13), the power spectrum of feature frequency F_0 will have a frequency component at feature frequency F_1. Hence, another criterion for selecting feature frequencies is that the two frequencies should not overlap in the frequency domain within half of the sampling frequency F_s. Note that the Fourier transform will smooth out frequency components higher than $F_s/2$.

D. Performance Metrics

We now discuss two metrics, detection rate and false positive rate, for evaluating the detection of a signal injected into a suspect circuit. Detection rate P_D is defined as the probability that *all bits of a signal* is correctly identified. The signal to noise ratio determines the probability that a feature frequency and the corresponding bit is identified. Denote the probability that feature frequency F_0 is recognized as p_{d0} and the probability that feature frequency F_1 is recognized as p_{d1}, respectively. Detection rate can be derived by,

$$P_D = p_{d0}^m p_{d1}^k, \tag{24}$$

where m is the number of 0 and k is the number of 1 in the signal. Because suspect connections may choose our exit and entry Tor routers simultaneously multiple n times, the overall detection rate after n times will be

$$P_{D,n} = 1 - (1 - P_D)^n. \tag{25}$$

When n approaches infinity, $P_{D,n}$ approaches 100%. This implies that if a Botnet continuously uses Tor, we will detect it sooner or later.

False positive rate P_F is the probability that there is no signal embedded into the traffic and the signal is incorrectly recovered from the traffic. Denote the probability that feature frequency F_0 appears in normal Tor traffic as p_{f0} and the probability that F_1 appears in normal Tor traffic as p_{f1}. The false positive rate can be derived as follows,

$$P_F = p_{f0}^m p_{f1}^k, \tag{26}$$

where m is the number of 0 and k is the number of 1 in the signal. Hence, by controlling the signal length, we can effectively reduce the false positive rate.

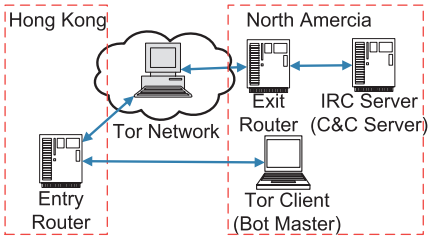

Fig. 17. Experiment setup for traceback.

VII. Experimental Evaluation of Traceback Approach

We have implemented the dual-tone multi-frequency signaling based traceback approach. Extensive real-world experiments were conducted to demonstrate the feasibility and effectiveness of our approach.

Figure 17 shows the experimental setup to evaluate the DTMF based traceback. We use PlanetLab [53] to deploy an entry router in Hong Kong. The exit router is deployed in USA, while the Tor client and IRC server are in Canada. The version of Tor in our experiments is 0.2.2.35. *mIRC* [54] is used as the IRC client of the botmaster and *UnrealIRCd* [46] emulates the IRC C&C server. By configuring the proxy setting of *mIRC*, we let *mIRC* communicate with the IRC server through the Tor network. Using the configuration file and manipulatable parameters, such as *EntryNodes*, *ExitNods*, *StrictEntryNodes*, and *StrictExitNodes* [55], we can control the client to choose both the entry and exit router along the circuit to carry out our experiments.

To evaluate the duel-tone multi-frequency signaling based traceback approach, we let the IRC client communicate with the emulated C&C server 30 times over Tor. At the Tor exit router, we choose two frequencies and control transmission frequency of Tor cells in order to embed our signal in the target traffic. At the entry onion router, the cells arriving at the circuit queue are recorded in a log file and the signal detection approach is applied to extract the feature frequency to recover a signal.

When we evaluate the false positive rate, the IRC client communicates with the emulated C&C 30 times through Tor again. Nonetheless, no signal is embedded into the traffic at the exit onion router. Denote the traffic without embedded signal as the clean traffic. We apply the detection approach to the clean traffic collected at the entry onion router. By checking whether a given signal is detected in the clean traffic, we obtain the false positive rate.

Figure 18 shows the relationship between frequency and amplitude in the frequency domain. In this set of experiments, the signal is 4-bits "1010". We adopt frequency 3Hz and 2Hz, or frequency 3Hz and 4Hz to encode bit "1" and bit "0", respectively. To extract these frequencies, we use the sampling frequency of 12Hz. In the upper row of Figure 18, frequency 3Hz is used for encoding bit "1", while frequency 2Hz for bit "0". We can clearly observe the high amplitudes at these feature frequencies to decode the specific signal bit "1010". In the second row of Figure 18, frequency 3Hz is used for encoding bit "1" and frequency 4Hz for bit "0", respectively.

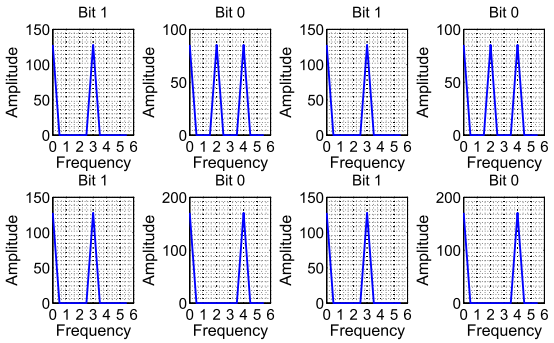

Fig. 18. Upper row: bits *1010* by feature frequencies 3 and 2 Hz; Bottom row: bits *1010* by feature frequencies 3 and 4 Hz.

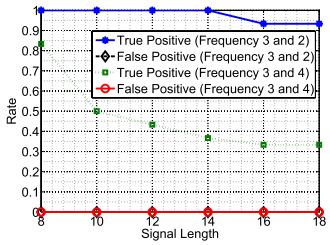

Fig. 19. Signal length (number of bits) versus rate.

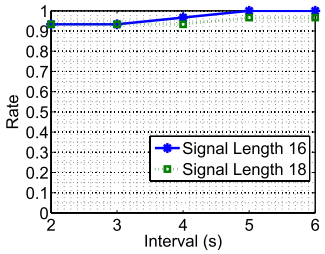

Fig. 20. Signal bit interval versus detection rate.

Likewise, we can identify the amplitudes at the feature frequencies using appropriate threshold λ. Using the decision rule in Section VI-B, we can recover the signal. Our results demonstrate that the DTMF approach works effectively.

Figure 19 illustrates the relationship between the detection rate and the signal length. As we can see from this figure, when the signal length is increased from 8 bits to 18 bits, the true positive rate will be slightly decreased by using frequencies 3Hz and 2Hz to embed the signal. When the signal length is between 8 bits and 14 bits, 100% positive rate can be achieved. However, when the signal length is 18 bits, the positive rate reduces to 92%. In addition, the positive rate when using frequencies 3Hz and 2Hz is much better than that of using frequencies 3Hz and 4Hz. It demonstrates that using high frequencies in DTMF may cause cell merging, which impacts the recovery of signals. This observation matches our analysis in Section VI. The false positive rate of these experiments approaches 0%, further validating the effectiveness of our traceback approach.

Figure 20 shows the relationship between the interval of signal bits and the detection rate. In this figure, we adopt the frequencies 3Hz and 2Hz to encode the signal, and test the

method with different signal lengths, i.e., 16 bits and 18 bits. We observe that when the interval of signal bits increases, the detection rate slightly increases. This observation also matches our analysis in Section VI.

VIII. Conclusion

In this paper, we presented a novel system, *TorWard*, for discovery, classification, and response of malicious traffic over Tor. In particular, *TorWard* inspects the passing traffic through an IDS at a Tor exit router while avoiding administrative and legal troubles by redirecting the traffic into Tor. We analyzed the data collected over a long period and discovered that a large amount of malicious traffic, including various P2P, botnet, spam, and other malware traffic, was carried over Tor. Among the 3, 624, 700 alerts recorded in one of our datasets, 78.03% of them are caused by P2P traffic, while 8.99% are related to malwares. To block malicious traffic at an exit router, we deployed IDS to forward alerts to a *sentinel* agent of *TorWard*, which can dynamically disrupt malicious traffic through our Tor control protocol. To facilitate forensic analysis, we designed an effective duel-tone multi-frequency (DTMF) signaling based approach to tracing malicious traffic across Tor. As an example that itself has significant practical meaning, we successfully traced the botnet traffic over Tor. The effectiveness and feasibility of *TorWard* were validated through a combination of extensive theoretical analysis and real-world experiments.

References

[1] The Tor Project, Inc. (2015). *Tor: Anonymity Online*. [Online]. Available: https://www.torproject.org/

[2] N. Christin, "Traveling the silk road: A measurement analysis of a large anonymous online marketplace," in *Proc. 22nd Int. World Wide Web Conf. (WWW)*, 2013, pp. 213–224.

[3] Z. Ling, J. Luo, K. Wu, and X. Fu, "Protocol-level hidden server discovery," in *Proc. 32nd IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2013, pp. 1043–1051.

[4] D. Storm. (2011). *Fingered by IP: Does It Take Chutzpah to Run a Tor Exit Relay?* [Online]. Available: http://blogs.computerworld.com/18892/fingered_by_ip_does_it_take_chutzpah_to_run_a_tor_exit_relay

[5] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the Tor network," in *Proc. 8th Int. Symp. Privacy Enhancing Technol. (PETS)*, 2008, pp. 63–76.

[6] A. Chaabane, P. Manils, and M. A. Kaafar, "Digging into anonymous traffic: A deep analysis of the Tor anonymizing network," in *Proc. 4th Int. Conf. Netw. Syst. Secur. (NSS)*, 2010, pp. 167–174.

[7] Z. Ling, J. Luo, K. Wu, W. Yu, and X. Fu, "TorWard: Discovery of malicious traffic over Tor," in *Proc. 33th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr./May 2014, pp. 1402–1410.

[8] D. Brown. (2010). *Resilient Botnet Command and Control With Tor*. [Online]. Available: https://www.defcon.org/images/defcon-18/dc-18-presentations/D.Brown/DEFCON-18-Brown-TorCnC.pdf

[9] (2015). *Tor2web: Visit Anonymous Websites*. [Online]. Available: http://tor2web.org/

[10] Anonymous. (2012). *IAmA a Malware Coder and Botnet Operator, AMA*. [Online]. Available: http://www.reddit.com/r/IAmA/comments/sq7cy/

[11] C. Guarnieri. (2012). *Skynet, a Tor-Powered Botnet Straight From Reddit*. [Online]. Available: https://community.rapid7.com/community/infosec/blog/2012/12/06/skynet-a-tor-powered-botnet-straight-from-reddit

[12] (2012). *Dec. 2012 Skynet Tor Botnet/Trojan.Tbot Samples*. [Online]. Available: http://contagiodump.blogspot.ca/2012/12/dec-2012-skynet-tor-botnet-trojantbot.html

[13] A. Matrosov. (2013). *The Rise of TOR-Based Botnets*. [Online]. Available: http://www.welivesecurity.com/2013/07/24/the-rise-of-tor-based-botnets/

[14] D. Danchev. (2013). *Cybercriminals Experiment With Tor-Based C&C, Ring-3-Rootkit Empowered, SPDY Form Grabbing Malware Bot*. [Online]. Available: http://blog.webroot.com/2013/07/02/cybercriminals-experiment-with-tor-based-cc-ring-3-rootkit-empowered-spdy-form-grabbing-malware-bot/

[15] L. Øverlier and P. Syverson, "Locating hidden servers," in *Proc. IEEE Secur. Privacy Symp. (S&P)*, May 2006, pp. 100–114.

[16] S. J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*, Nov. 2006, pp. 27–36.

[17] L. Zhang, J. Luo, M. Yang, and G. He, "Application-level attack against Tor's hidden service," in *Proc. 6th Int. Conf. Pervasive Comput. Appl.*, 2011, pp. 509–516.

[18] A. Biryukov, I. Pustogarov, and R.-P. Weinmann, "Trawling for Tor hidden services: Detection, measurement, deanonymization," in *Proc. 34th IEEE Symp. Secur. Privacy (S&P)*, May 2013, pp. 80–94.

[19] G. Tian, Z. Duan, T. Baumeister, and Y. Dong, "A traceback attack on freenet," in *Proc. 32th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Apr. 2013, pp. 1797–1805.

[20] (2015). *Freenet*. [Online]. Available: https://freenetproject.org/

[21] Y. Xiang, I. Natgunanathan, D. Peng, W. Zhou, and S. Yu, "A dual-channel time-spread echo method for audio watermarking," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 383–392, Apr. 2012.

[22] S. Yu, G. Zhao, W. Dou, and S. James, "Predicted packet padding for anonymous Web browsing against traffic analysis attacks," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 4, pp. 1381–1393, Aug. 2012.

[23] (2015). *Transparently Routing Traffic Through Tor*. [Online]. Available: https://trac.torproject.org/projects/tor/wiki/doc/TransparentProxy

[24] Open Information Security Foundation (OISF). (2015). *Suricata*. [Online]. Available: http://www.openinfosecfoundation.org/

[25] Emerging Threats Pro, LLC. (2015). *Emerging Threats*. [Online]. Available: http://www.emergingthreats.net/

[26] (2015). *Barnyard2*. [Online]. Available: http://www.securixlive.com/barnyard2/

[27] (2008). *Basic Analysis and Security Engine (BASE) Project*. [Online]. Available: http://base.secureideas.net/

[28] Z. Ling, J. Luo, W. Yu, M. Yang, and X. Fu, "Extensive analysis and large-scale empirical evaluation of Tor bridge discovery," in *Proc. 31th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, Mar. 2012, pp. 2381–2389.

[29] (2015). *Tor Network Status*. [Online]. Available: http://torstatus.blutmagie.de/

[30] (2015). *nDPI—Open and Extensible LGPLv3 Deep Packet Inspection Library*. [Online]. Available: http://www.ntop.org/products/ndpi/

[31] (2015). *TShark*. [Online]. Available: http://www.wireshark.org/docs/man-pages/tshark.html

[32] (2015). *Tor on Android*. [Online]. Available: https://www.torproject.org/docs/android.html.en

[33] M. Tigas. (2015). *Onion Browser*. [Online]. Available: https://mike.tig.as/onionbrowser/

[34] K. Dunham. (2007). *The Russian Business Network*. [Online]. Available: https://www.issa.org/Library/Journals/2007/July/Dunham%20-%20RiskRadar%20-%20The%20Russian%20Business%20Network.pdf

[35] (2015). *DShield*. [Online]. Available: http://www.dshield.org/

[36] (2015). *Spamhaus*. [Online]. Available: http://www.spamhaus.org/

[37] D. Gerzo. (2012). *Brute Force Blocker*. [Online]. Available: http://danger.rulez.sk/projects/bruteforceblocker/

[38] (2015). *OpenBL.org—Abuse Reporting and Blacklisting*. [Online]. Available: http://www.openbl.org/

[39] (2015). *CI Army*. [Online]. Available: http://www.ciarmy.com/

[40] (2015). *SNORT Users Manual 2.9.5, Snort Default Classifications*. [Online]. Available: http://manual.snort.org/node31.html#Snort_Default_Classifications

[41] (2015). *Shadowserver*. [Online]. Available: https://www.shadowserver.org/wiki/

[42] Fortinet. (2011). *A Guide to SpyEye C&C Messages*. [Online]. Available: http://blog.fortinet.com/a-guide-to-spyeye-cc-messages/

[43] (2015). *Mt.Gox*. [Online]. Available: https://www.mtgox.com/

[44] U. Eriksson. (2015). *Pen*. [Online]. Available: http://siag.nu/pen/

[45] (2015). *TSOCKS—A Transparent SOCKS Proxying Library*. [Online]. Available: http://tsocks.sourceforge.net

[46] (2015). *UnrealIRCd*. [Online]. Available: http://www.unrealircd.com/

[47] Sourcefire, Inc. (2015). *Sourcefire Vulnerability Research Team (VRT)*. [Online]. Available: http://www.snort.org/vrt

[48] The Tor project, Inc. (2015). *Tor Control Protocol Specification*. [Online]. Available: https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=control-spec.txt

[49] (2015). *Dual-Tone Multi-Frequency Signaling*. [Online]. Available: http://en.wikipedia.org/wiki/Dual-tone_multi-frequency_signaling
[50] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in *Proc. IEEE Secur. Privacy Symp. (S&P)*, May 2006, pp. 183–195.
[51] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against anonymous systems," in *Proc. ACM Workshop Privacy Electron. Soc. (WPES)*, Oct. 2007, pp. 11–20.
[52] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia, "A new cell-counting-based attack against Tor," in *Proc. 16th ACM CCS*, Nov. 2009, pp. 578–589.
[53] (2015). *PlanetLab: An Open Platform for Developing, Deploying, and Accessing Planetary-Scale Services*. [Online]. Available: http://www.planet-lab.org/
[54] mIRC Co. Ltd. (2015). *mIRC: Internet Relay Chat Client*. [Online]. Available: http://www.mirc.com/
[55] R. Dingledine and N. Mathewson. (2015). *Tor Path Specification*. [Online]. Available: https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt

Zhen Ling received the B.S. degree from the Nanjing Institute of Technology, China, in 2005, and the Ph.D. degree in computer science from Southeast University, China, in 2014. He was with the Department of Computer Science, City University of Hong Kong, from 2008 to 2009, as a Research Associate, and the Department of Computer Science, University of Victoria, from 2011 to 2013, as a Visiting Scholar. He is currently an Assistant Professor with the School of Computer Science and Engineering, Southeast University, Nanjing, China. His research interests include network security, privacy, and forensics.

Junzhou Luo (M'07) received the B.S. degree in applied mathematics and the M.S. and Ph.D. degrees in computer network from Southeast University, Nanjing, China, in 1982, 1992, and 2000, respectively. He is currently a Full Professor with the School of Computer Science and Engineering, Southeast University. His research interests are next generation network, protocol engineering, network security and management, cloud computing, and wireless LAN. He is a member of the Association for Computing Machinery, and the Cochair of the IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design.

Kui Wu (SM'07) received the B.Sc. and the M.Sc. degrees in computer science from Wuhan University, China, in 1990 and 1993, respectively, and the Ph.D. degree in computing science from the University of Alberta, Canada, in 2002. He joined the Department of Computer Science, University of Victoria, Canada, in 2002, where he is currently a Professor. His research interests cover network performance analysis, online social networks, Internet of Things, and parallel and distributed algorithms.

Wei Yu received the B.S. degree in electrical engineering from the Nanjing University of Technology, Nanjing, China, in 1992, the M.S. degree in electrical engineering from Tongji University, Shanghai, China, in 1995, and the Ph.D. degree in computer engineering from Texas A&M University, in 2008. He is currently an Associate Professor with the Department of Computer and Information Sciences, Towson University. His research interests include cyberspace security, computer networks, and cyber-physical systems. He received the U.S. National Science Foundation Early Career Award in 2014.

Xinwen Fu received the B.S. degree in electrical engineering from Xian Jiaotong University, China, in 1995, the M.S. degrees in electrical engineering from the University of Science and Technology of China, in 1998, and the Ph.D. degree in computer engineering from Texas A&M University, College Station, in 2005. He is currently an Associate Professor with the Department of Computer Science, University of Massachusetts Lowell. His current research interests include network security and privacy, digital forensics, wireless networks, and network QoS. His research was featured on CNN and China Central Television.