TorWard: Discovery of Malicious Traffic over Tor

Zhen Ling*†, Junzhou Luo*, Kui Wu†, Wei Yu‡ and Xinwen Fu§
*Southeast University, Email: {zhenling, jluo}@seu.edu.cn
†University of Victoria, Email: wkui@cs.uvic.ca
‡Towson University, Email: wyu@towson.edu
‡University of Massachusetts Lowell, Email: xinwenfu@cs.uml.edu

***Abstract*—Tor is a popular low-latency anonymous communication system. However, it is currently abused in various ways. Tor exit routers are frequently troubled by administrative and legal complaints. To gain an insight into such abuse, we design and implement a novel system, *TorWard*, for the discovery and 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 IDS (Intrusion Detection System) is used to discover and classify malicious traffic. We performed comprehensive analysis and extensive real-world experiments to validate the feasibility and effectiveness of *TorWard*. Our data shows that around 10% Tor traffic can trigger IDS alerts. Malicious traffic includes P2P traffic, malware traffic (e.g., botnet traffic), DoS (Denial-of-Service) attack traffic, spam, and others. Around 200 known malware have been identified. To the best of our knowledge, we are the first to perform malicious traffic categorization over Tor.**

***Keywords*—Tor, Malicious Traffic, Intrusion Detection System.**

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]. Tor has been growing and consists of around 3800 volunteer Tor routers as of July 2013. 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 servers (C&C) 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 or even raided by police [4]. 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 and classification. Our major contributions are summarized as follows.

TorWard can be deployed on *a university campus* while avoiding 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. Nevertheless, this is the only safe way to investigate malicious traffic over Tor in such 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 in order not to interfere with non-Tor traffic. Since the use of *TorWard* in early 2012, we have not received any complaints in our experiments, while a lot of administrative complaints were received each day with a bare exit router on campus. We also perform theoretical analysis to demonstrate the effectiveness of *TorWard* and the real-world data matches our theoretical analysis well.

With *TorWard*, we conduct statistical analysis of malicious traffic through Tor. Key observations include: 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

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

978-1-4799-3360-0/14/$31.00

we did not manually filter out all false alarms given the huge volume of traffic, we give confirmed examples of major threats such as botnet traffic and 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. We also derived *traffic protocol and application* statistics, which is largely consistent with the study in [5], [6] while we now can observe traffic from mobile devices. To the best of our knowledge, this is the first paper to perform malicious traffic categorization over Tor.

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 theoretically 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. We conclude this paper in Section V.

II. Background and Related Work

In this section, we briefly introduce Tor and related work.

A. Tor

Figure 1 illustrates the basic architecture of the Tor network. It consists of four components: Tor client, onion routers, directory servers, and application server. Generally speaking, a Tor client installs *onion proxy* (*OP*) that is an interface between Tor network and clients. *Onion routers* (*OR*) form the core Tor network and relay traffic between Tor client and 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 server over Tor, the client downloads onion router information from directory servers and chooses a series of onion routers to establish a three-hop path[2], referred to as *circuit*. The three onion routers are known as *entry* (OR1), *middle* (OR2), and *exit* onion router (OR3), respectively. The client can mix multiple TCP connections, referred to as *streams*, over a single Tor circuit.

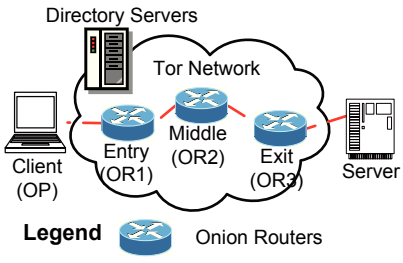

Fig. 1. Tor Network

[2]3 is the default value in Tor.

B. Related Work

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

Malware authors have began to make use of Tor to provide anonymous communication between malware and C&C servers. For example, Dennis Brown [7] showed how to configure Zeus bot through Tor2Web [8] 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 [9]. 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 [10], [11]. Two Tor hidden service based malwares [12], [13] were reported in July 2013.

Research has been performed to discover Tor hidden servers [3], [14]–[17]. For example, Øverlier and Syverson [14] proposed the packet counting based traffic analysis to identify a hidden server at entry onion routers. Zhang *et al.* [16] leveraged HTTP features to identify a hidden server at entry onion routers. Murdoch [15] employed a clock skew based approach to check whether a given Tor node is a hidden server or not. Ling *et al.* [3] proposed a protocol-level based hidden server discovery approach. Biryukov *et al.* [17] studied how to deploy the hidden service directory to harvest hidden service information and investigated the packet counting based traffic analysis to locate hidden servers.

Other anonymous communication systems were widely abused as well as Tor. For example, Tian *et al.* [18] studied how to trace back the receiver who is retrieving illegal file over the Freenet [19].

III. Malicious Traffic Collection

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

We categorize Tor traffic 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. Therefore, 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 2 illustrates the structure of this system. *TorWard* consists of four logical components: a firewall, an IDS, a transparent proxy, and a Tor exit router. The first three components are actually hosted at a NAT gateway machine. We built a private network and setup a firewall on the gateway that connects the

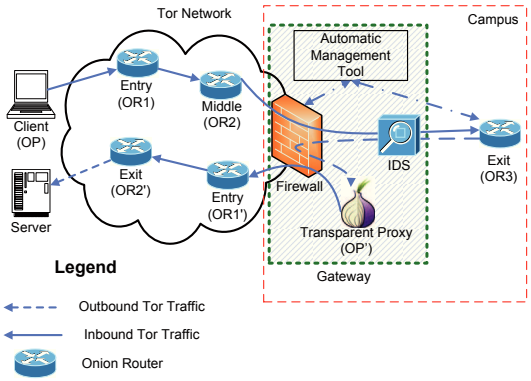

Fig. 2. System Architecture for Malicious Traffic Collection

private network to the campus network. The private network includes a Tor exit router and a Tor client. Port forwarding is enabled at the firewall to enable communication between the exit router and middle routers in the public network. To attract 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 [20] 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 3 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. 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. The IDS, *Suricata* [21], is

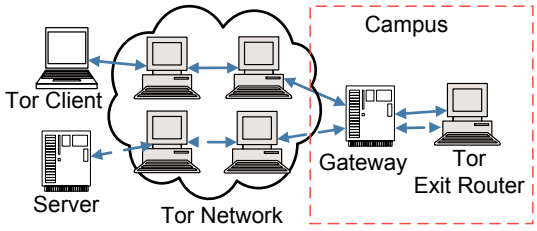

Fig. 3. Experiment Setup for Malicious Traffic Discovery

deployed on the gateway. The IDS detects the destination of outbound traffic from the Tor exit router and uses signature based rules to detect potential malicious traffic.

We adopt the IDS rules from Emerging Threats [22] for *Suricata*. Note that *Suricata* records alerts into unified binary files, and *Barnyard2* [23] 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* [24] is deployed as a GUI to display alerts stored in the database.

C. *Effectiveness of* TorWard

To demonstrate the effectiveness of *TorWard* and justify we can use a few or even one exit router to derive reliable traffic statistics over Tor, we perform 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 [25], 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. Let n be 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

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

We can see that $\mathcal{P}_n$ grows significantly as n increases. According to the current Tor router bandwidth real-world data [26], 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 4 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.

Let $\mathcal{P}_k(b)$ be the probability that a malicious Tor client creates at least one circuit traversing our exit router after establishing k circuits. Assume that $\mathcal{P}_k(b)$ approaches 100%. Let t_i be the average time of creating a new circuit for the i^{th} type of malicious Tor client. We can obtain the average of total time T of retrieving all m malicious traffic by

$$T = max\{t_1 * k, \ldots, t_i * k, \ldots, t_m * k\}. \quad (3)$$

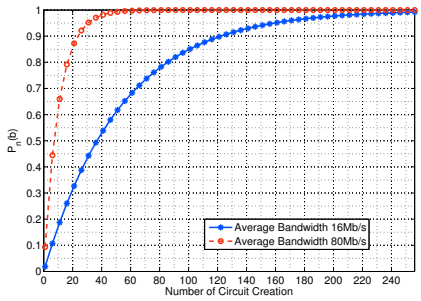

Fig. 4. P_n vs number of Circuits

According to Equation (3), the average lifetime of a circuit can influence the total time T for retrieving all m types of malicious traffic. Denote T_t and T_c as the lifetime of the connection created by Tor client and the lifetime of a circuit, respectively. Tor circuits can be classified into two types: dirty circuit and clean circuit [27]. A dirty circuit is attached to at least one stream, while a clean circuit never carries a stream. According to the Tor protocol, T_c can be computed as $max\{10\ minutes, T_t\}$. The lifetime of a clean circuit is 60 minutes. Consequently, if a malicious Tor client builds a short-term connection with a remote server through Tor, a new circuit will be established every 10 minutes, which will increase the chance of being discovered by our exit router.

For example, a typical HTTP based bot should periodically establish a nonpersistent connection to the C&C server in order to report the information retrieved from the victim computer. Hence, this type of malicious Tor client will continuously switch the circuits, and frequent switching will dramatically increase the number of circuits created. Some malicious Tor client might create a long-term connection with a remote server through Tor. For example, an IRC based bot will build a persistent connection with remote server so as to receive commands from the bot master. In this case, it will take much more time to discover malicious Tor clients. However, the natural Tor routers churn [28] will force the Tor client to use a new circuit. In addition, a victim host might go offline regularly, forcing the Tor client to create new circuits.

We can infer the average time of discovering a malicious Tor client in terms of the average lifetime of a circuit. According to existing research results [27], the distribution of the lifetime of a circuit is a long tailed distribution. The lifetime of around 1.5% circuits exceeds 2 hours, and the average lifetime of a circuit is around 200 seconds. In Figure 4, when k is equal to 260, the probability $\mathcal{P}_{260}(16)$ approaches 100%. Accordingly, it takes an average time of around $200 * 260/3600 \approx 14$ hours for our exit router to be selected at least one time by a malicious Tor client. We need $2 * 260/24 \approx 22$ days to capture the malicious client using a long term circuit. Since our experiments ran for over a month, the derived statistics can reliably reflect the current status of malicious traffic over Tor.

TABLE I. DATASETS

Dataset	IDS Ruleset	Period	Size (TB)
Dataset 1	ETOpen	Oct. 03, 2012 $\sim$ Nov. 12, 2012	3.95
Dataset 2	ETPro	Jun. 12, 2013 $\sim$ 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

IV. 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.

A. Discovered Traffic

We conducted our experiments with *TorWard* in Figure 3 during two periods: October 3, 2012 to November 12, 2012, and June 12, 2013 to July 17, 2013. During the two periods, we observed a large amount of traffic, denoted as dataset 1 and dataset 2, respectively. Table I describes the two datasets. We applied a free version of IDS ruleset, i.e., Emerging Threats ETOpen [22], to obtain alerts for dataset 1. To discover more malicious traffic, a commercial product of IDS ruleset, i.e., Emerging Threats ETPro [22], was used to obtain alerts for dataset 2. We observed similar traffic patterns in the two datasets, and we will focus on dataset 2 in this paper to save space.

We apply the deep packet inspection library *nDPI* [29] to dataset 2 to derive traffic protocol statistics. As we mentioned in Section III-A, the traffic traversing our Tor exit node consists

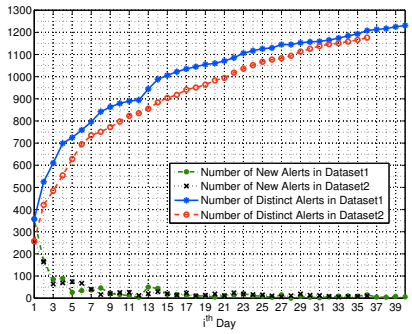

Fig. 5. The relation between number of discovered alerts and ith day

of inbound and outbound Tor traffic. To identify inbound Tor traffic, we use TShark's protocol filter [30] 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 [6], [31]. The P2P traffic is the source of various copyright infringement issues and is the reason why a Tor exit node is bombarded with various DMCA (Digital Millennium Copyright Act) 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 [6], [31] 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, 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 [21], 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. Below 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.

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	

Misc-attack alerts are generated for blacklisted hosts. The IDS rules contain IP addresses of hosts or netblocks that 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 [22].

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.

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.

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

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 form 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. Since 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, and 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. 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 that attempts to utilize 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 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 5, 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 5.

```
GET /outlawz/mainp/gate.php?guid=GTIGT-
FDCCD9A7405DI30457F77&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. 6. Spyeye checkin

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. Note 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 calculate the volume of incoming and outgoing traffic from raw data based on the alerts. Tables VI and VII show the traffic information of dataset 1. Tables VIII and IX 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 $(16GB + 375GB) = 391GB$ out of $3.95TB$ traffic, i.e., around 10% traffic in dataset 1, can trigger the alerts. Moreover, 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.

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.

TABLE VI. INCOMING TRAFFIC STATISTICS (DATASET 1)

Classification	Size (Bytes)	Percentage	Priority
Shellcode-detect	3,880,691,862	24.36%	High
Policy-violation	350,720,319	2.20%	High
Trojan-activity	147,539,256	0.93%	High
Web-application-attack	18,419,483	0.12%	High
Attempted-user	4,974,891	0.03%	High
Attempted-admin	683,681	0.004%	High
Bad-unknown	155,986,606	0.98%	Medium
Misc-attack	11,186,217	0.07%	Medium
Attempted-recon	174	0.000001%	Medium
Misc-activity	46,947,883	0.29%	Low
Not-suspicious	35,000,208	0.22%	Low
Network-scan	1,018	0.000006%	Low
Unclassified	11,278,737,267	70.80%	Unknown
Total	15,930,888,865		

TABLE VII. OUTGOING TRAFFIC STATISTICS (DATASET 1)

Classification	Size (Bytes)	Percentage	Priority
Policy-violation	370,283,402,491	98.70%	High
Trojan-activity	3,932,683,916	1.05%	High
Attempted-user	142,657	0.00003%	High
Web-application-attack	122,106	0.00003%	High
Bad-unknown	730,210,376	0.19%	Medium
Attempted-recon	72,195,354	0.02%	Medium
Not-suspicious	80,125,728	0.02%	Low
Misc-activity	50,643,140	0.01%	Low
Total	375,149,525,768		

TABLE VIII. 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		

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. Therefore, 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 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 6, a Spyeye bot is connecting to its C&C server to report the information of the victim machine. According to the

TABLE IX. 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		

```
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. 7. Ngrbot

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 7 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 8 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 X, 89.02% alerts originate from only 10 countries, while around 50% 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.

```
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. 8. Mobile Malware (Android/Plankton.P)

TABLE X. 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

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., create a new block to show a proof of work. Currently, 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 [43]. Nevertheless, 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 [10], [11] 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.

V. CONCLUSION

In this paper, we present a novel system, *TorWard*, for malicious traffic monitoring over Tor. *TorWard* can explore the passing traffic through an IDS at a Tor exit router while avoiding administrative and legal troubles by redirecting the traffic

into Tor. We analyze the data collected over a long period and discover that Tor carries a large amount of malicious traffic, including various P2P, botnet, spam, and other malware traffic. Among the $3,624,700$ alerts raised in one of our datasets, 78.03% of them are caused by P2P traffic, while 8.99% are related to malwares. As an ongoing work, we are conducting research on blocking and sanitizing malicious traffic at Tor exit routers and contributing to the healthy development of Tor.

ACKNOWLEDGMENTS

This work was supported in part by National Key Basic Research program of China under grants 2010CB328104, China National High Technology Research and Development Program under Grants No. 2013AA013503, National Natural Science Foundation of China under grants 61272054, 61202449 and 61320106007, Natural Sciences and Engineering Research Council of Canada (NSERC), by US NSF grants 1116644, 0942113, 0958477, 0943479, and 1117175, China National Key Technology R&D Program under Grants No. 2010BAI88B03 and 2011BAK21B02, by China Specialized Research Fund for the Doctoral Program of Higher Education under grants 20110092130002, Science Research Foundation of Graduate School of Southeast University, Jiangsu Provincial Key Laboratory of Network and Information Security under grants BM2003201, and Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under grants 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.

REFERENCES

[1] The Tor Project, Inc., "Tor: Anonymity Online," https://www.torproject.org/, 2013.
[2] N. Christin, "Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace," in *Proceedings of the 22nd International World Wide Web Conference (WWW)*, 2013.
[3] Z. Ling, J. Luo, K. Wu, and X. Fu, "Protocol-level Hidden Server Discovery," in *Proceedings of the 32th IEEE International Conference on Computer Communications (INFOCOM)*, 2013.
[4] Darlene Storm, "Fingered by IP: Does it take chutzpah to run a Tor exit relay?" http://blogs.computerworld.com/18892/fingered_by_ip_does_it_take_chutzpah_to_run_a_tor_exit_relay, 2011.
[5] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining Light in Dark Places: Understanding the Tor Network," in *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies (PETS)*, 2008.
[6] A. Chaabane, P. Manils, and M. A. Kaafar, "Digging into Anonymous Traffic: A Deep Analysis of the Tor Anonymizing Network," in *Proceedings of the 4th International Conference on Network and System Security (NSS)*, 2010.
[7] Dennis Brown, "Resilient Botnet Command and Control with Tor," https://www.defcon.org/images/defcon-18/dc-18-presentations/D.Brown/DEFCON-18-Brown-TorCnC.pdf, 2010.
[8] "Tor2web: Visit Anonymous Websites," http://tor2web.org/, 2013.
[9] Anonymous, "IAmA a malware coder and botnet operator, AMA," http://www.reddit.com/r/IAmA/comments/sq7cy/, 2012.
[10] Claudio Guarnieri, "Skynet, a Tor-powered botnet straight from Reddit," https://community.rapid7.com/community/infosec/blog/2012/12/06/skynet-a-tor-powered-botnet-straight-from-reddit, 2012.
[11] "Dec. 2012 Skynet Tor botnet / Trojan.Tbot samples," http://contagiodump.blogspot.ca/2012/12/dec-2012-skynet-tor-botnet-trojantbot.html, 2012.
[12] Aleksandr Matrosov, "The rise of TOR-based botnets," http://www.welivesecurity.com/2013/07/24/the-rise-of-tor-based-botnets/, 2013.
[13] Dancho Danchev, "Cybercriminals experiment with Tor-based C&C, ring-3-rootkit empowered, SPDY form grabbing malware bot," http://blog.webroot.com/2013/07/02/cybercriminals-experiment-with-tor-based-cc-ring-3-rootkit-empowered-spdy-form-grabbing-malware-bot/, 2013.
[14] L. Øverlier and P. Syverson, "Locating Hidden Servers," in *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, 2006.
[15] S. J. Murdoch, "Hot or Not: Revealing Hidden Services by Their Clock Skew," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, November 2006.
[16] L. Zhang, J. Luo, M. Yang, and G. He, "Application-level attack against Tor's hidden service," in *Proceedings of the 6th International Conference on Pervasive Computing and Applications*, 2011.
[17] A. Biryukov, I. Pustogarov, and R.-P. Weinmann, "Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization," in *Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P)*, 2013.
[18] G. Tian, Z. Duan, T. Baumeister, and Y. Dong, "A Traceback Attack on Freenet," in *Proceedings of the 32th IEEE International Conference on Computer Communications (INFOCOM)*, 2013.
[19] "Freenet," https://freenetproject.org/, 2013.
[20] "Transparently Routing Traffic Through Tor," https://trac.torproject.org/projects/tor/wiki/doc/TransparentProxy, 2013.
[21] Open Information Security Foundation (OISF), "Suricata," http://www.openinfosecfoundation.org/, 2013.
[22] Emerging Threats Pro, LLC., "Emerging Threats," http://www.emergingthreats.net/, 2013.
[23] "Barnyard2," http://www.securixlive.com/barnyard2/, 2013.
[24] "Basic Analysis and Security Engine (BASE) project," http://base.secureideas.net/, 2008.
[25] Z. Ling, J. Luo, W. Yu, M. Yang, and X. Fu, "Extensive Analysis and Large-Scale Empirical Evaluation of Tor Bridge Discovery," in *Proceedings of the 31th IEEE International Conference on Computer Communications (INFOCOM)*, 2012.
[26] "Tor Network Status," http://torstatus.blutmagie.de/, 2013.
[27] A. Biryukov, I. Pustogarov, and R.-P. Weinmann, "TorScan: Deanonymizing Connections Using Topology Leaks," in *Proceedings of the 17th European Symposium on Research in Computer Security (ESORICS)*, 2012.
[28] T. Elahi, K. Bauer, M. AlSabah, R. Dingledine, and I. Goldberg, "Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor," in *Proceedings of the 11th ACM Workshop on Privacy in the Electronic Society (WPES)*, 2012.
[29] "nDPI - Open and Extensible GPLv3 Deep Packet Inspection Library," http://www.ntop.org/products/ndpi/, 2013.
[30] "TShark," http://www.wireshark.org/docs/man-pages/tshark.html, 2013.
[31] D. Mccoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining Light in Dark Places: Understanding the Tor Network," in *Proceedings of the 8th Privacy Enhancing Technologies Symposium (PETS)*, 2008.
[32] "Tor on Android," https://www.torproject.org/docs/android.html.en, 2013.
[33] Mike Tigas, "Onion Browser," https://mike.tig.as/onionbrowser/, 2013.
[34] Ken Dunham, "The Russian Business Network," https://www.issa.org/Library/Journals/2007/July/Dunham%20-%20RiskRadar%20-%20The%20Russian%20Business%20Network.pdf, 2007.
[35] "DShield," http://www.dshield.org/, 2013.
[36] "Spamhaus," http://www.spamhaus.org/, 2013.
[37] Daniel Gerzo, "Brute Force Blocker," http://danger.rulez.sk/projects/bruteforceblocker/, 2012.
[38] "OpenBL.org - Blacklisting and Abuse Reporting," http://www.openbl.org/, 2013.
[39] "C.I.Army," http://www.ciarmy.com/, 2013.
[40] SNORT Users Manual 2.9.5, "Snort Default Classifications," http://manual.snort.org/node31.html#Snort_Default_Classifications, 2013.
[41] "shadowserver," https://www.shadowserver.org/wiki/, 2013.
[42] Fortinet Blog, "A Guide to SpyEye C&C Messages," http://blog.fortinet.com/a-guide-to-spyeye-cc-messages/, 2011.
[43] "Mt.Gox," https://www.mtgox.com/, 2013.