Int. J. Inf. Secur. (2015) 14:205–220
DOI 10.1007/s10207-014-0256-7

REGULAR CONTRIBUTION

Detection and analysis of eavesdropping in anonymous communication networks

Sambuddho Chakravarty · Georgios Portokalidis · Michalis Polychronakis · Angelos D. Keromytis

Published online: 18 August 2014

Abstract Anonymous communication networks, like Tor, partially protect the confidentiality of user traffic by encrypting all communications within the overlay network. However, when the relayed traffic reaches the boundaries of the network, toward its destination, the original user traffic is inevitably exposed to the final node on the path. As a result, users transmitting sensitive data, like authentication credentials, over such networks, risk having their data intercepted and exposed, unless end-to-end encryption is used. Eavesdropping can be performed by malicious or compromised relay nodes, as well as any rogue network entity on the path toward the actual destination. Furthermore, end-to-end encryption does not assure defense against man-in-the-middle attacks. In this work, we explore the use of decoys at multiple levels for the detection of traffic interception by malicious nodes of proxy-based anonymous communication systems. Our approach relies on the injection of traffic that exposes bait credentials for decoy services requiring user authentication, and URLs to seemingly sensitive decoy documents which, when opened, invoke scripts alerting about being accessed. Our aim was to entice prospective eavesdroppers to access our decoy servers and decoy documents, using the snooped credentials and URLs. We have deployed our prototype implementation in the Tor network using decoy IMAP, SMTP, and HTTP servers. During the course of over 30 months, our system has detected 18 cases of traffic eavesdropping that involved 14 different Tor exit nodes.

Keywords Tor · Anonymity networks · Proxies · Eavesdropping · Decoys

S. Chakravarty (✉) · M. Polychronakis · A. D. Keromytis
Columbia University, New York, USA
e-mail: sc2516@cs.columbia.edu

M. Polychronakis
e-mail: mikepo@cs.columbia.edu

A. D. Keromytis
e-mail: angelos@cs.columbia.edu

G. Portokalidis
Stevens Institute of Technology, Hoboken, USA
e-mail: gportoka@stevens.edu

1 Introduction

Many services and protocols rely on non-encrypted communication. Consequently, malicious users or organizations that have access to the network elements through which user traffic is routed can eavesdrop and obtain sensitive data, such as user authentication credentials. This situation can potentially worsen when users employ proxy-based systems to access the same services without using end-to-end encryption, as the number of hosts or nodes that can eavesdrop on their traffic increases. Various public and private networks may block access to social networking and other popular online services for various reasons. Under these conditions, users often resort to using distributed proxying systems to prevent their traffic from being filtered. They resort to such mechanisms so as to evade network traffic filtering based on source, destination, and content.

Anonymous communication systems [1,2,22,31,44] are popular examples of proxy-based systems, which enable users to hide their IP address from the services they use, and often employ encryption by design. Systems such as Tor [22] route data through a series of proxies. Data packets are encrypted multiple times [16], so that if adversaries intercept the traffic en-route to the destination, they will not be able to determine the actual source or destination of the traffic. The process also aids in achieving confidentiality against eavesdropping adversaries who can observe the traffic and

snoop out sensitive and reusable information such as user names, passwords, and HTTP cookies.

In such systems, however, the last node on a path can access the original message that is being transmitted to the intended recipient. Many users are not aware of this discrepancy between the anonymity and privacy guarantees offered by these systems, and the lack of end-to-end data confidentiality which is often mistakenly assumed. Disregarding the absence of end-to-end confidentiality, users often send sensitive information through these relays. Some of these relays, acting with malicious intent, may misuse sensitive user information such as user names and passwords, URLs to sensitive information, and HTTP session cookies. Thus, in exchange for anonymity, users place their trust in components of the anonymous communication system that could potentially abuse it. In all cases, user data at some point is available in their original form. McCoy et al. [34] have shown that there are Tor exit nodes which indeed eavesdrop on the traffic flowing through them, abusing users' trust.

An obvious solution to such problems might involve sending traffic encrypted using SSL through relays. However, malicious relay operators can employ man-in-the-middle attacks and snoop on the traffic of even SSL-encrypted sessions [51], and attacks of this kind have been observed in the Tor network [4].

Our approach for the detection of misbehaving relay nodes involves the transmission of decoy traffic that contains easily reusable and seemingly sensitive information (such as fake plain-text user names and passwords) via all nodes of the anonymization network to decoy servers under our control. Relay nodes eavesdropping on user traffic may try to reuse this information and connect to our decoy servers. In this paper, we present the overall architecture of our eavesdrop detection system, which can be used to detect eavesdropping by untrusted nodes of various anonymization systems (and proxying systems in general). We have implemented our system for detecting eavesdropping by malicious Tor exit nodes. Tor is among the most widely used relay-based anonymization networks, with over half a million worldwide users [53]. Our system could, however, be easily adapted for other relay-based anonymization networks as well, such as JAP [31] and I2P [28].

The use of fake information, or *honeytokens* [46], for detecting unauthorized access to sensitive data, has been explored previously for several applications related to network intrusion detection and misbehavior detection. However, there has not been adequate research done in using such information and systems to detect misbehavior by otherwise trusted nodes of anonymization enabling networks and systems. McCoy et al. [34] were the first to explore the use of transmitting decoy TCP traffic through Tor nodes to see which nodes eavesdrop on them. Their approach required access to DNS traffic and could be trivially defeated by an adversary by simply using appropriate command line options of tools such as `tcpdump` [33] which by default perform reverse DNS look up for unresolved IP addresses. Our approach, in contrast, neither requires access to DNS traffic nor relies on default behavior of traffic capture tools. We send innocuous-appearing TCP traffic through Tor exit nodes exposing easily reusable decoy information and periodically check the server logs for subsequent unsolicited connection attempts. The decoy information being unique to each Tor exit helps in easily identifying the Tor exit node involved in the eavesdropping incident. Section 2 extensively describes all related research endeavors.

In previous work [15], we described how we could use the system to detect eavesdrop using plain-text IMAP and SMTP protocol messages, exposing fake usernames and passwords to Tor exit nodes. Our previous effort describes various eavesdropping incidents detected between August 2010 and May 2011. Thereafter, based on the activities of various adversaries who logged into our system using the exposed user credentials and on ideas and concepts borrowed from various related research efforts (such as using decoy documents to detect insider attacks; proposed by Bowen et al. [10–12]), we have extended our system with the following components:

Honeypots Based on the detected activities of some eavesdropping incidents (described later in Sect. 3) we have added SSH and FTP honeypots to the system.

Beacon-bearing decoy documents We deployed a web server which hosts decoy documents, generated using D^3 [10,13], presenting fake but alluring information such as fake credit card numbers, and usernames and passwords to fake `paypal.com` accounts. These documents contain *beacons* that are triggered when the documents are opened, connecting to a remote site and reporting information such as time and IP address of the host from which the document was accessed. The URLs to these decoy documents are regularly exposed to Tor exit nodes through HTTP GET and POST messages with the hope that potential eavesdropper would reuse the exposed URLs to access the decoy documents.

Apart from the above additions, we have also explored how our system could be used to detect more advanced eavesdropping incidents, such as HTTP cookie hijacking and SSL MITM attacks. We describe these in Sect. 5.

As a proof of concept, we have deployed various decoy servers and honeypots, and transmitted decoy traffic to these systems via all Tor exit nodes. Our system has been operational for over 30 months. During this period, it has detected 18 incidents of eavesdropping by Tor exit nodes (some of which have been previously described in our related previous work [15]). In some cases, the adversary connected back to our decoy server from the same exit node itself while in

others the adversary connected back from other exit nodes or hosts in other networks. Often, an eavesdropping exit node stopped serving Tor traffic soon after the eavesdropping incident was detected and in one case even reappeared after several months.

In summary, the main contributions of this paper are the following:

- An architecture for detecting various forms of traffic snooping by nodes of anonymous communication networks (and proxy servers in general) that involves the exposure of reusable decoy information, such as plain-text user credentials and URLs to sensitive appearing documents containing beacons.
- A prototype of our proposed system using various decoy servers and honeypots that has been deployed in the Tor anonymous communication network.
- A detailed forensic analysis of the eavesdropping incidents recorded by our system.

2 Background information

2.1 Anonymous network communication systems

Anonymous network communication systems enable users to hide their identity from their communication peers. Most of these systems rely on sending traffic via one or more proxies and may additionally encrypt traffic, using concepts presented by Chaum [16], to obfuscate the true source or destination of messages. Such systems are often classified as *low-latency* and *high-latency* anonymous communication systems. Low-latency systems are designed to be efficient for *semi*-interactive applications such as web browsing and instant messaging. High-latency systems are geared toward delay tolerant applications such as e-mail. Low-latency network anonymization systems are further classified based on the routing paradigms they employ—those that are derived from onion routing [21], and those that are based upon Crowds [44]. Systems such as Tor [22], JAP [31], and I2P [28] employ deterministic routing, wherein the set of proxies through which the traffic is sent is known by the connection or session initiator. Systems such as GNUNet [8], BitBlender [7], and OneSwarm [30] employ probabilistic traffic routing schemes similar to Crowds. Each traffic forwarding relay in such a system randomly chooses to send the traffic either to the destination or to another relay in the system.

Tor Tor [22] is closely modeled on the onion routing [43] paradigm and is one of the most widely used low-latency anonymity networks, with an estimated user base of more than 500,000 users (as of March 2013 [52]). Tor aims to protect the anonymity of Internet users by relaying user TCP streams through a network of overlay nodes run by volunteers. Tor also supports responder anonymity through *hidden services*.[1]

The Tor overlay network consists of over 2,500 proxies, known as *onion routers* (ORs), which are mostly operated by volunteers scattered across the globe. User traffic is relayed through *circuits*, which are formed by persistent TLS connections between different nodes. By default, Tor circuits consist of three nodes: the first one is known as the *entry node*, the second one as the *middleman*, and the third is called the *exit node*. During circuit establishment, a Tor client negotiates shared secret keys with the relays that it chooses for the circuit. Thereafter, the client uses these keys to encrypt the transmitted messages in multiple layers of encryption, starting with the public key of the exit node. Each of the nodes then first "peels off" one layer of encryption and then forwards the message to the next node on the circuit. The exit node decrypts the final layer of encryption, which reveals the original plain-text message of the user, and forwards it to its actual destination through a regular TCP connection. Thus, if in-transit traffic is intercepted by eavesdroppers, they cannot determine the actual source or destination of the traffic.

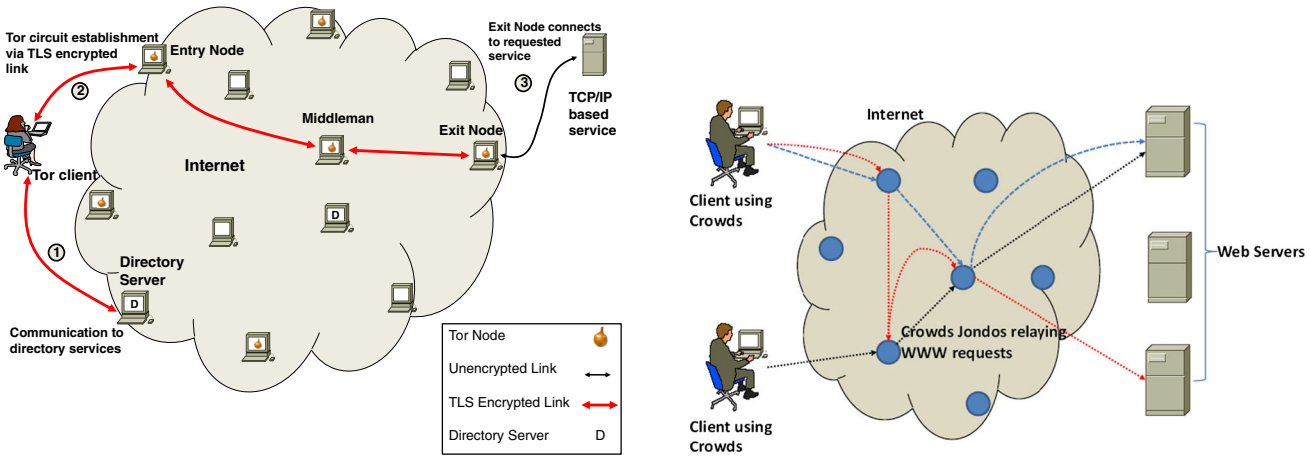
Figure 1a presents the basic steps for the creation of a new Tor circuit consisting of three onion routers:

1. The Tor client queries the directory service to obtain a list of the available Tor relays.
2. The client uses a set of relays to create Tor circuits. By default, circuits are created using three relays.
3. The client selects one of the circuits and creates a TCP connection to its entry node. Traffic is forwarded through the circuit to the exit node, which communicates directly with the actual destination.

Tor also provides configurable access control features to exit node operators. Usually, Tor exit nodes are configured to allow traffic forwarding for only a small set of TCP services. The supported services are defined by the operator of the exit node through the specification of an *exit policy*.

Crowds Reiter et al. designed Crowds [44] for anonymous web browsing. Crowds is based on the principle of probabilistic forwarding. Like Tor (and other onion routing systems), it relies on an overlay network for enabling responder anonymity. The nodes of the overlay network are called *jondos*. A web browser using this system forwards a web request to a randomly chosen jondo. Upon receiving the request, the

[1] A TCP-based service can keep its IP address hidden (and thus its identity) by replacing the IP address with a hidden service URL. These URLs end in a virtual top-level domain called ".onion" and are resolved by a Tor clients while initiating connection to the hidden service.



(a) **(b)**

Fig. 1 Overview of different anonymity networks based on routing paradigms. **a** Basic steps for communicating through Tor. The client obtains a list of the available Tor relays from a directory service ①, establishes a circuit using multiple Tor nodes ②, and then starts forwarding its traffic through the newly created circuit ③. **b** Crowds users establish paths via Jondos to web services

jondo forwards the request to another jondo with probability p_f or sends it directly to the intended web service. Figure 1b schematically represents the functioning of the crowds. Many of modern anonymity preserving P2P file-sharing systems such as GNUNet [9], BitBlender [7], and OneSwarm [30] are derived from Crowds. We do not focus further on Crowds paradigm we have implemented and demonstrated our architecture for the Tor network.

2.2 Network and system misbehavior detection

Our work is closely related to research efforts that involve the exposure of enticing decoy information or resources to lure potential adversaries, with the objective of identifying them and their modus operandi. One of the first uses of decoy information for enabling the observation of real malicious activity has been documented by Clifford Stoll [48]. In his book, *The Cuckoo's Egg* [49], the author recounts his efforts to trap an intruder that broke into the systems of the Lawrence Berkeley National Laboratory. As part of his efforts to monitor the actions and trace the intruder's origin, he generated fake documents containing supposedly classified information that would lure the intruder to come back and stay longer on the compromised computer.

Computer-based systems and resources deployed widely with the objective of luring prospective adversaries and intruders for logging their identities and actions are widely known as *honeypots* [41,47]. Such systems have no production value other than being compromised and subsequently aid in tracking the actions of the attacker. Honeypots have been extensively used for modeling, logging, and analyzing attacks originating from sources not only external to an network [27,56], but also from within its perimeter [13].

Complementary to honeypots, researchers and system administrators often use *honeytokens* [46] which are pieces of information with purpose no other than being intercepted or stolen and abused by an adversary. Any use of these honeytokens clearly indicates unauthorized access. The decoy credentials used in our approach, which we describe in detail in the next section, can be classified as a variety of honeytokens.

More recently, Bowen et al. [11] proposed the use of decoy documents to detect misbehaving entities within the perimeter of an organization. The decoy documents contained embedded "beacons," such as scripts or mac-ros, which are executed when the document is opened. The authors used fake tax records bearing information appearing to be "sensitive" and enticing to an adversary. In case a document is opened, the embedded beacon connects to an external host and transmits information such as time of access and IP address of the hosts used to open the document.

In another related research work, Bowen et al. [12] used real WiFi traffic as a basis for the generation of decoy traffic with realistic network interactions. An API is used to insert bait content, such as popular webmail service cookies, FTP and HTTP messages, and so on into these decoy packets. The packets were then broadcasted through an unencrypted Wi-Fi network and exposed to potential eavesdroppers. Unsolicited connection attempts to the services, using the bait credentials, are marked as illegitimate. In their experiments, the authors replayed `gmail.com` and `PayPal.com` messages carrying credentials and cookies for decoy accounts, and utilized the last login IP address feature of these services for deter-

mining illegitimate connection attempts. Such techniques are not applicable anymore as the aforementioned popular web mail and financial services encrypt their connections with SSL.

There has been little effort in detecting misbehaving overlay nodes of anonymity networks. In a work most closely related to ours, McCoy et al. [34] attempted to detect eavesdropping on malicious Tor exit routers by taking advantage of the IP address resolution functionality of network traffic capturing tools. Packet sniffing tools such as `tcpdump` [33] are by default configured to resolve the IP addresses of the captured packets to their respective DNS names. Their system transmitted, via Tor exit nodes, TCP SYN packets destined to unused IP addresses in a block owned by the system's operator. When the packet capturing program attempted to resolve the IP address of a probe packet, it issued a DNS request to the authoritative DNS server. The system's operator had access to the traffic going to this authoritative DNS server. Thus, requests to this DNS server with the unused IP address were an indication that probe packets had been intercepted by some packet capturing program and could be traced back to the network host where they were captured. However, when capturing traffic on disk, `tcpdump` by default does not resolve any addresses, and in any case the eavesdropper can trivially disable this functionality, rendering the above technique ineffective.

In contrast to that work, our system does not require access to the DNS server traffic. We employ decoy clients and servers which communicate via Tor exits and present easily reusable sensitive appearing information such as user credentials and URLs to sensitive appearing documents which also contain beacons such as those described above. The system periodically monitors the client and server logs for unsolicited connection attempts to the server which are marked as suspicious. Our system is flexible enough to be adapted to detect eavesdropping in various different protocols. As a proof of concept, we have augmented the system we presented in our previous paper [15] with an SSH honeypot and decoy FTP and HTTP servers presenting decoy documents containing beacons. Additionally, we explored detecting HTTP cookie hijack from traffic going to social network sites and HTTPS man-in-the-middle attacks by malicious exit node operators.

3 System architecture

In this section, we present the architecture of our traffic eavesdropping detection system that we have deployed for Tor. We describe the design of the decoy traffic transmission mechanism and the corresponding decoy services, as well as the approach we used for incident data collection and correlation.

3.1 Approach

Eavesdropping on network traffic is a passive operation without any directly observable effects. However, the fact that some traffic has been intercepted can be potentially inferred, when a third party that should not have access to the intercepted data uses it. For example, an eavesdropper can steal user credentials for services that do not use application-layer encryption, such as user names and passwords for Web sites with poor user authentication implementations, or for servers that use clear-text sign-in protocols, such as FTP or IMAP. Thereafter, any attempt by the eavesdropper to access the user's account is an observable event. The problem with the latter lies in (a service) identifying whether the use of a set of credentials was made by a third party or the user.

Our approach is based on the assumption that an eavesdropper will use the intercepted data in some manner. We use two types of decoy data that are not used in any other way, to determine with certainty that data was intercepted by a proxy. First, we use decoy authentication credentials to decoy services, essentially honeypots, under our control. The use of these credentials with our services at a later time is a clear indication that eavesdropping occurred. Second, we transmit URLs to decoy documents containing information of potential value, such as decoy `PayPal.com` accounts, and fake financial transactions including fake credit card information. Later downloads of these documents also indicate eavesdropping. Every decoy is uniquely transmitted through exactly one proxy, so that we can later associate its use with it.

We further exploit the fact that the eavesdropper will probably attempt to open these documents to collect further information. We use `D-Cubed` [10] to automatically generate the PDF and Microsoft Word documents and embed *beacons*, basically scripts, into them. These documents present information about fake `Pay-Pal.com` and credit card accounts, to lure adversaries. These scripts get launched automatically, when the documents are viewed by the eavesdropper, connecting to a remote host under our control, and transmitting information such as the IP address of the host used to open these documents. This aids us in gathering more information about adversaries, e.g., their geographic location based on their IP address.

Figure 2 illustrates the overall design of our system when applied on the Tor network. A client under our control periodically connects through Tor to our decoy server and transmits easily reusable clear-text information, such as user authentication credentials. As a result, such easily re-usable and potentially sensitive information is exposed to exit nodes of Tor circuits (and any other network entity between the exit node and the decoy server). Both the client and the server record detailed information about any attempted connection (such as user credentials, URLs, and connection time stamps). These logs are thereafter periodically tal-

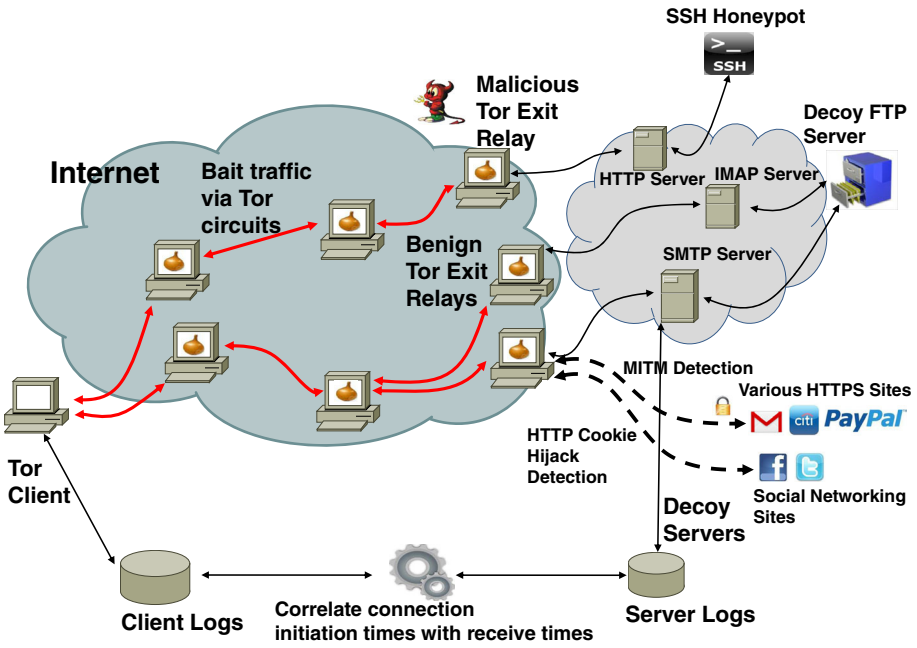


Fig. 2 Overall architecture of the proposed traffic interception detection system when applied on the Tor network

lied to determine unsolicited connection attempts, which are marked as suspicious.

In more detail, as the system is continuously running, the following steps take place periodically:

1. The client connects to the decoy server through Tor and sends information such as unique user authentication credentials (in case of IMAP and SMTP decoy servers) in clear-text, and URLs for sensitive appearing decoy documents containing beacons (in case of HTTP decoy server) through clear-text HTTP GET and POST protocol messages. The client creates circuits through *all* the exit nodes. Using unique user credentials and URLs per exit node helps in identifying the actual exit node involved when eavesdropping is detected.
2. The decoy server maintains detailed record for each session that may include the user name and password (for IMAP and SMTP), the IP address of the exit node used in the connection, the URL pointing to the unique decoy documents, and the time stamp corresponding to when the connection to the decoy server was established.
3. After a successfully completed session on the decoy server, the system attempts to correlate it with a recently completed client session. Connections observed on the server for which there are no corresponding client connection attempts are labeled as suspicious.

Each of the unique user credentials and URLs to decoy documents is associated with exactly one exit node and is exposed to it through a Tor circuit terminating at that node. Thus, the exit node involved in a particular eavesdropping incident is known based on the given set of credentials or URLs used in the unsolicited session observed by the decoy server.

Our system has been optimized compared to our initial prototype [15] for gathering more information regarding the adversaries' activities. Attempts to reuse some of the IMAP decoy credentials with other services, like FTP and SSH, urged us to also set up honeypots with decoy FTP and SSH services running. Moreover, we used the FTP server to also serve decoy documents. We describe all incidents in detail and present up-to-date information for new and previously detected eavesdroppers [14], in Sect. 4.

Note that our approach can be adapted to detect more advanced traffic interception attacks. In Sect. 5, we describe an extension that can detect HTTP cookie hijacking attacks and man-in-the-middle attacks by malicious Tor exit nodes.

3.2 Implementation

Although Tor can forward the traffic of any TCP-based network service, in practice not all exit routers support all application protocols. For example, SMTP relay through port 25 is blocked by the majority of Tor exit nodes to prevent spammers from covertly relaying their messages through the Tor network. Consequently, the first important decision we had to take before beginning the implementation of our prototype system was to choose a set of services that are supported by a large number of Tor exit nodes. At the same time, candidate services should support unencrypted authentication through a clear-text protocol, while the services themselves should entice potential eavesdroppers.

Tor exit nodes are usually configured to allow traffic forwarding or only a small set of TCP services. The ser-

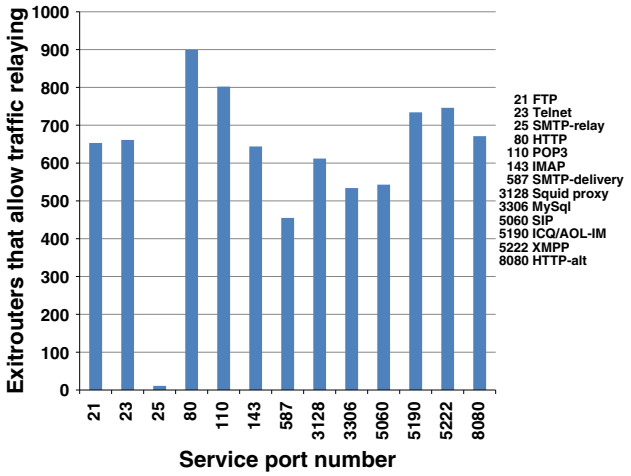


Fig. 3 Number of Tor exit nodes that allow traffic relaying through different TCP port numbers, for services that support clear-text protocols

vices allowed are defined by the operator of the exit node through the specification of an exit policy. To determine the most widely supported unencrypted application protocols, we queried the Tor directory servers and retrieved the number of exit nodes that allowed each different protocol. Figure 3 presents the number of Tor exit nodes that at the time of the experiment allowed the relaying of traffic through various TCP port numbers. In accordance with the results obtained by McCoy et al. [34], widely used applications such as web browsing, email retrieval, and instant messaging are allowed by a large number of exit nodes. We found approximately 900 exit nodes that allowed access to port 80. We also found 644 exit nodes supporting exit to IMAP (port 143) and 455 exit nodes supporting SMTP delivery (port 587). Both these protocols support plain-text user authentication. They involve transmission of plain-text usernames and passwords.[2]

Credentials for accessing users' messages that may contain sensitive private information, or for sending e-mails through verified user addresses, can be of high value for a malicious eavesdropper. This led us to choose the IMAP and SMTP protocols for our prototype implementation. Furthermore, due to the wide prevalence of exit nodes that allow exit for web traffic, we decided to use HTTP GET and POST messages to expose URLs of decoy documents containing beacons, stored on a decoy web server that we control.

3.2.1 Decoy traffic transmission and eavesdropping detection

Our decoy traffic transmission subsystem is based on a custom client that supports the IMAP and SMTP protocols. The client has been implemented using Perl, and service protocol emulation is provided by the `Net::IMAPClient` and `Net::SMTP` modules. We use `curl` [24] for transmitting the HTTP GET and POST messages to expose the URL of decoy documents to the exit nodes. The clients and servers are hosted on Intel x86 based machines running Ubuntu Linux.

Every day, for each service, our Tor client connects to the decoy service several times via circuits through each of the exit nodes. This is achieved by establishing a new Tor circuit for each connection, and enforcing each circuit to use a particular exit node. Once a connection has been established, the client authenticates on the server using a unique set of credentials associated with the particular combination of exit node and decoy server.[3] Thereafter, the client performs activities, such as browsing through some folders in case of IMAP, or sending a fake e-mail message in case of SMTP, so as the protocol message exchanges appear realistic. In case some exit node is not accessible, the corresponding set of credentials is skipped. Similarly, when a new exit node joins the overlay network, a new set of credentials for each decoy service is generated for use only with that exit node. To achieve this, the Tor directory services are periodically queried and fresh list of exit nodes supporting exit to the requisite service (IMAP or SMTP delivery) is retrieved. Thereafter, new user credentials are assigned to the set of fresh exit nodes.

Usernames are generated as a combination of names in various languages [38] and random numbers using `language confluxer` [39] and `prop` [40]. Passwords for these usernames are generated using `pwgen` [54].

Similar to the decoy IMAP and SMTP client processes, a routine sends and retrieves decoy documents to and from a decoy web server, through circuits via each Tor exit. The process exposes URLs of the decoy documents carrying the beacons, to the exit node through HTTP POST and GET messages. Each exit node is associated with a set of unique decoy documents which are sent to and received from the server, through HTTP POST and GET messages, respectively. We assume rogue exit nodes to be snooping on HTTP traffic and accessing the exposed decoy document URLs.

Under normal operational conditions, the number of connections successfully initiated by the client each day, through each exit node, should equal to the number of connections received by the server from each of these exit nodes. For each successfully initiated connection, the client-side scripts log information such as exit node involved decoy credentials used and the connection start and end times. The corresponding information for these connections, as seen at the

[2] In contrast to SMTP relay (port 25), SMTP through port 587 is dedicated to message submission for delivery only for users that have registered accounts on the server.

[3] In other words, for each exit node that allows access to IMAP, we created a unique username and password. This unique association of the exit node and the exposed user credential helps identify the eavesdropping exit nodes that snoop on these exposed credentials and connect back to our decoy server.

server, is obtained from the server's logs. Any unsolicited successful connection using some of the previously transmitted decoy credentials is labeled as an illegitimate suspicious connection attempt. Such suspicious connections are identified by tallying the connections initiated by our client to those received by the server, based on the logs recorded at the client and the server. Specifically, upon the completion of a successful connection, the decoy server sends directly (not through Tor) to the client all the recorded information about the recently completed session. The client then compares the connection details, including the set of credentials, decoy documents, the exit node involved, and the start and end times of the connections recorded by both the client and the server, against the recently completed connections. For each connection successfully initiated by the client, the system checks whether the corresponding connection can be identified from the server logs. Since the system knows the exit node involved each connection, and the fact that it is used with exactly one set of decoy credentials, it is expected that for each connection initiated by the client, there is exactly one connection arriving at the server that matches the said connection attempt. In case no matching connection is found, the system generates a report that includes the time of the last generated connection that used the intercepted credentials, the time of the unsolicited connection to the server, the IP address of its initiator, and the exit node involved in the incident.

3.2.2 Important implementation considerations

During the implementation of our prototype system, we dealt with various issues related to improving the accuracy of our traffic interception detection approach, or with cases where interesting design trade-offs came up. We briefly describe some of these issues in the rest of this section.

Quality of decoy traffic and honeypot services The believability of the decoy traffic [12] is a crucial aspect of the effectiveness of our approach. For instance, a decoy IMAP session using an account that does not have a realistic folder structure, or that does not contain any real e-mail messages, might raise suspicions to an eavesdropper. Repeating the same actions in every session, or launching new sessions at exactly the same time every day, can also be indications that the sessions are artificially generated. In our prototype system, we vary the connection times and activity in each session, and we use realistically looking folder structures for the IMAP accounts and send innocuous-appearing e-mail messages. The inboxes of these decoy accounts contain messages attached with decoy documents containing the beacons, generated from the `D-Cubed` system, presenting enticing information such as decoy `PayPal.com` accounts and fake financial transactions involving fake credit card numbers. These documents are attached to e-mail messages containing banking jargon to reduce suspicion.

As mentioned above, in some of the eavesdropping incidents, the adversaries actually tried to access other services such as SSH and FTP using the exposed IMAP credentials. Thus, we installed a FTP server, hosting user accounts corresponding to each of the IMAP users. Each of these accounts used the same passwords, which were used the IMAP accounts. The users' FTP directories were also populated with decoy documents containing the beacons. Further, to make these accounts appear innocuous, we also placed documents taken from [45] and source code documentations and help files taken from an open source program.

To track the behavior of adversaries who may try to log in to SSH accounts using the exposed IMAP credentials, we installed `kippo` [18], an open-source, medium-interaction, and easy to configure SSH honeypot, on the virtual machine hosting our FTP server. `Kippo` supports multiple users and thus seemed well suited for our setup consisting of several hundred fake user accounts and passwords. These fake SSH user accounts of `kippo` used the same usernames credentials as those used for the IMAP and FTP accounts. The IMAP server redirects all FTP and SSH requests to this virtual machine hosting the SSH honeypot and the decoy FTP server.

Time Synchronization Accurate time synchronization between the client and the decoy server(s) ensures proper correlation of the connections generated by the client with the connections received by the server, and the correct identification of any unsolicited connections. Although the volume of our decoy connections is very low, allowing any illegitimate connections to easily stand out, the clocks of all hosts in our architecture are kept synchronized using the Network Time Protocol. The sub-second accuracy of NTP allows the precise correlation of the connection start and end times observed on both the client and server. This offers an additional safeguard for the verification of the detected traffic interception incidents.

Eavesdropping Incident Verification Besides the accurate correlation between the start and end times, logged by the client and the server, we have taken extra precautions to avoid any inaccurate classification of our generated decoy connections as illegitimate. For each connection launched by the client, the system also keeps track of the circuit establishment times by monitoring Tor client's control port. Moreover, we have enabled all the built-in logging mechanisms provided by the Tor software. On the server side, all the incoming and outgoing network traffic is captured using `tcpdump`. In addition to the server logs, the captured traffic provides valuable forensic information regarding the nature of illegitimate connections, such as the exact sequence of protocol messages sent by the attacker's IMAP, SMTP, and HTTP clients.

4 Deployment results

Our prototype implementation has been continuously operational in the Tor network since August 2010. During the course of over 30 months of its operation, our system has detected sixteen traffic interception incidents. In this section, we describe the eavesdropping and subsequent malicious connection attempts using the snooped user credentials. We analyze the consequent activities of the intruders as they were recorded in the decoy server logs. Our incident description Web site [14] contains information about the exit nodes involved in each incident and details of the activities of the intruders once they logged in our decoy server using the snooped user credentials.

4.1 Eavesdropping incidents

The observed eavesdropping incidents were related to different exit nodes, and all the related illegitimate connections were received by our decoy IMAP server. Based on the intercepted credentials used in each unsolicited connection, we were able to identify the Tor exit node involved in each incident. Information about the detected incidents (*e.g.*, date, exit node location, and activities recorded by the server) is presented in Table 1. The detail of the remaining incidents are available in our Web site [14].

While most of the incidents involved a different exit node, there were some, such as one in India and another one in South Korea, which eavesdropped on exposed credentials repeatedly and connected back to our decoy server. The ones in India, hosted in the same ISP network, repeatedly connected back to our decoy server for weeks. Even after we modified the passwords associated with the IMAP accounts, the exit operators learnt the modified passwords by snooping on the traffic and connected back to the decoy server with these new passwords. The node in South Korea eavesdropped on our decoy traffic three times. Although after each connect-back attempt the exit node was inaccessible, it surfaced after sometime and again attempted to eavesdrop on the decoy traffic.

There were also several incidents in which the malicious exit nodes were not accessible for days after the eavesdropping incidents and subsequent connect-back attempts. Also, as evident from Table 1, in the majority of the incidents, the adversaries connect to the decoy server via other exit nodes or hosts, probably as an attempt to hide their true identities. However, in the first four incidents, which occurred together, the connect-back attempts originated directly from the exit nodes at which the decoy user credentials were exposed. All these connect-back attempts occurred within four to 6 h after the exposure of the decoy credentials, an interval significantly shorter compared to the rest of the incidents. We thus speculate that these first four eavesdropping cases were coordinated by the same individual or group, probably using the same tools or methodology in each case.

Ten eavesdropping incidents were detected between August 2010 and April 2011. These have been described in detail in our previous paper [15]. Eight new incidents were detected between November 2011 and March 2012. By and large, in each of these incidents, the modus operandi of the adversaries was similar to that of the previous incidents.

The first of these incidents that occurred in November 2011 involved an exit node in the UK. The exit node operator connected back to our server via a host that was running in a cloud service provider's network. Thereafter, in several instances, the involved exit node was inaccessible soon after the eavesdropping incident.

The second incident (the 12th one in Table 1) involved an exit node in Russia. In this, the adversary tried to connect to an SSH server using the exposed credentials, which prompted us to install an SSH honeypot. All SSH connections arriving to the IMAP server, involving the decoy credentials, were directed to the SSH honeypot.

The next incident involved an exit node in South Korea which had eavesdropped earlier in September 2010 and connected to the decoy server via another exit node. Thereafter, it was inaccessible for a considerable period of time after which it resurfaced and eavesdropped again.

The fourth incident involved an exit node in Germany. The adversary connected back to the decoy server several times via an exit node in the Netherlands. In each of these connect-back attempts, the adversary connected to our decoy server for a few minutes and logged out. After analyzing the exchanged protocol messages, it seems that the adversary connected to the server without using any known mail client program (perhaps using their own custom mail client that simply connects and disconnects, thereby checking the validity of the credentials).[4]

The fifth one involved an exit node in the Netherlands. The exit node operator connected back to our decoy server via an exit node in Switzerland. The malicious exit node operator connected to our decoy server only once for a short duration and then logged out.

The sixth incident involved an exit node in the USA. The exit node operator eavesdropped and subsequently connected back to the decoy server via an exit node in Canada. The adversary connected several times to the decoy server, using an email client program (possibly `Kmail`).

[4] Mail clients generally execute a set of commands on the server to fetch the various user directories associated with an account. The absence of such commands and zero payload length could be a strong indication that the adversary does not use any known mail client. We have studied the various protocol messages exchanged by various popular mail client programs.

Table 1 Observed traffic interception incidents during first 21 months of the deployment

Incident number	Date	Exit node location	Remarks
1	Aug.'10	US	Same pattern as in incidents 2, 3, and 4
			Connect-back from the same exit node
2	Aug.'10	Hong Kong	Same pattern as in incidents 1, 3, and 4
			Connect-back from the same exit node
3	Aug.'10	UK	Same pattern as in incidents 1, 2, and 4
			Connect-back from the same exit node
4	Aug.'10	The Netherlands	Same pattern as in incidents 1, 2, and 3
			Connect-back from the same exit node
5	Sep.'10	S. Korea	Connect-back from a different exit node
6	Sep.'10	Hong Kong	Connect-back from a third party host
			Exit node not accessible upon detection
7	Sep.'10	India	Connect-back from third-party hosts
			Exit node not accessible upon detection
8	Jan.'11	Germany	Connect-back from third-party hosts
			Attempt to use SSL through the IMAP
			`STARTTLS` command
9	Apr.'11	India	Connect-back from third-party hosts and other Tor relays
10	Apr.'11	India	Same as 9. Both exit nodes in the same
			ISP network and many of the third-party connect-back hosts were in the same networks (mostly in Europe and India)
			Was involved in incident 7
11	Nov. '11	UK	Connect back via a host in a web-hosting and cloud service providing organization
12	Nov. '11	Russia	Connect back via another host in a Russian ISP
13	Nov. '11	S. Korea	Exit node involved in incident 5
14	Jan. '12	Germany	Connect back via another Tor exit in The Netherlands
15	Jan. '12	The Netherlands	Connect back via another Tor exit in Switzerland
16	Jan. '12	US	Connect back via another host in a Canadian ISP
17	Jan. '12	S. Korea	Exit node involved in incidents 5 and 13
18	Mar. '12	Estonia	Connect back via another host in a Polish ISP

In all cases, the eavesdropper connected to our decoy IMAP server using a set of intercepted decoy credentials

The next one involved the exit node in South Korea, the one whose operator had eavesdropped twice previously. Each time the exit node operator connected back to our server, the exit node became inaccessible for several weeks and resurfaced, only to be detected again by our system. After this seventh incident, the exit node was finally blacklisted by Tor's operators.

Finally, the eighth incident (the 18th one listed in Table 1) involved an exit node in Estonia, whose operator eavesdropped and connected back to our decoy server via an exit node in Poland. There was only one connect-back attempt by the adversary that lasted for about 10 min.

Figure 4 presents this time differences between the exposure of the decoy credentials and the subsequent connect-back attempts for each of the incidents. The horizontal axis represents the eavesdropping and connect-back events. The vertical axis denotes the time delay between the exposure

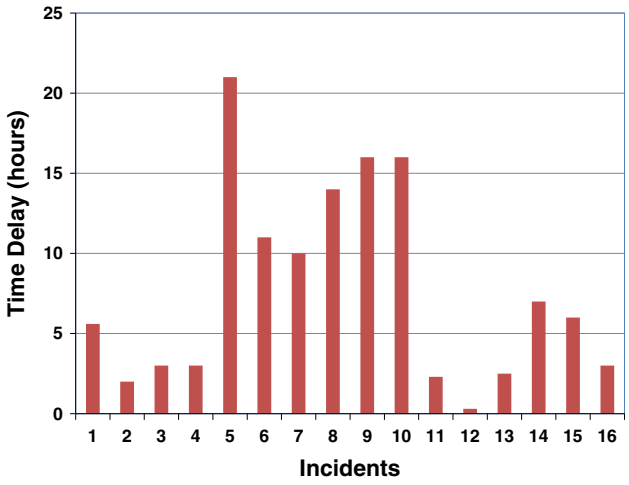


Fig. 4 Time difference between the exposure of the decoy credentials and the first connect-back attempt on the decoy server

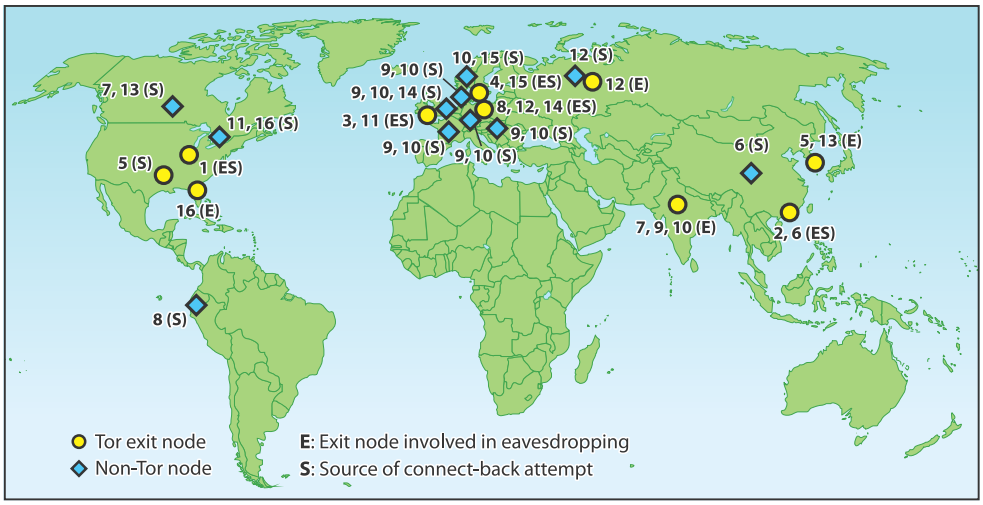


Fig. 5 Locations of the Tor exit nodes involved in the observed traffic interception incidents, and the non-Tor hosts that connected back to the decoy servers. Numbers refer to the corresponding incidents listed in Table 1

of the decoy credential and its subsequent usage in connect-back attempts.

The map in Fig. 5 presents an overall view of the geographic locations of the exit nodes and the third-party hosts involved in the observed incidents. Tor and non-Tor nodes are represented using different symbols. We used basic geo-IP address lookup tools which provide only country-level accuracy, so the points on the map denote only the country in which each host was located. The number next to each point corresponds to the incident number, as presented in Table 1.

Apart from detecting eavesdropping on plain-text protocol messages, our system could be used to detect more advanced attacks. In the next section, we demonstrate how our system could be used to detect SSL man-in-the-middle attacks. We detected an exit node in Russia that was involved in such attacks.

4.2 Adversaries' activities

In some of the incidents, the adversary connected to the decoy server using popular e-mail clients, while in the rest, they connected directly to the server and manually issued protocol command messages. Popular e-mail clients issue a certain default set of commands to access the mail folders such as `INBOX`, `Drafts`, `Sent`, and so on. The IMAP protocol allows users to issue various commands that can result in fetching the contents of these folders. Each mail clients issues a somewhat different set of commands to fetch the contents of these folders. The commands, and their order in which they are issued, can be treated as a "signature" for the client. We analyzed the signatures of various popular e-mail clients (*e.g.*, `Thunderbird`, `MS Outlook`, `Balsa` [5], `Claw` [17], `Sylpheed` [50], `Kmail` [32] and `Evolution` [25]) to determine the possible e-mail clients the adversaries used. From the network traffic, corresponding to the time when the adversaries connected to the decoy server using IMAP clients, it appears that `Kmail`, `Evolution`, and `Thunderbird` were commonly used for connecting to our decoy server.

In the incidents where the adversary connected manually to the IMAP server, we observed the adversary executing various different kinds of commands. In one, adversaries connected and switched to TLS mode so as to hide their activities. We thereafter turned off the capability in the server to switch to TLS mode after establishing connections. In another incident, the adversary issued esoteric IMAP4 ACL [35] commands. In yet others, the adversary tried to use credentials to log into services such as FTP and SSH. These services were, however, inaccessible using the decoy user credentials. These activities compelled us to install the decoy FTP server and SSH honeypot. The IMAP server redirects the connection attempt to FTP and SSH services to the decoy FTP server and SSH honeypot so as to lure the attackers to the download decoy documents or try to execute programs from the terminal interface, thereby aiding in gathering further information about such attackers.

4.3 Volume of decoy traffic injected

For each connection to the IMAP decoy server and subsequent protocol message exchanges, we sent approximately 15.4 KB of traffic. For 644 decoy accounts, each corresponding to a unique exit node, this number totals to approximately 10 MB. In case of such connections and exchanges to the

Table 2 Available bandwidth of the malicious exit nodes as reported by http://torstatus.blutmagie.de/

Incident number	Advertised Bandwidth	Remarks
1	Unknown	Relay was not running when accessed
2	Unknown	Relay was not running when accessed
3	44 Mbit/s	Guard node with high uptime
4	20.8 Mbit/s	Guard node with high uptime
5	1.4 Mbit/s	Advertises high uptime
6	56 Kbit/s	Advertises high uptime, runs directory service
7	Unknown	Relay was not running when accessed
8	856 Kbit/s	Guard node with high uptime, runs directory service
9	150 Kbit/s	Non-guard exit node
10	100 Kbit/s	Non-guard exit node
11	Unknown	Relay was not running when accessed
12	1 Mbit/s	Guard node with relatively high uptime
13	320 Kbit/s	Non-guard exit node
14	8.5 Mbit/s	Non-guard exit node with high bandwidth and uptime
15	Unknown	Relay was not running when accessed
16	336 Kbit/s	Non-guard exit node
17	320 Kbit/s	Non-guard exit node
18	1.6 Mbit/s	Guard node with relatively high uptime

SMTP decoy server, we sent about 21.3 KB of traffic. For 455 decoy SMTP accounts, this figure comes out to be approximately 9.7 MB.[5]

4.4 Attributes of the exit nodes involved in the incidents

Table 2 shows the available bandwidth of the exit nodes that were involved in the detected incidents. Two of the exit nodes advertised very high available bandwidth (44 and 20.8 Mbit/s, respectively) and thus are very likely to be selected in Tor client circuits, as the default Tor circuit node selection mechanism is biased toward nodes with high advertised available bandwidth [20]. There were some which advertised somewhat lesser bandwidth of 8.5 and 1.4 Mbit/s. Finally, there were some which advertised yet lower bandwidths of less than 1 Mbit/s. Both of the high bandwidth, and two of the lower bandwidth nodes, were *guard nodes*[6] with high up-times.

[5] This difference is primarily due to the different lengths of IMAP and SMTP messages. The overhead due to Tor protocol messages, involving circuit setup, key exchanges, accounting, and circuit termination, does not vary significantly between IMAP and SMTP.

[6] By default, a fixed set of entry nodes used by Tor clients to defend against traffic analysis attacks that can be launched by malicious entry and exit nodes.

5 Other efforts and possibilities: HTTP session cookie hijack and SSL MITM detection

Apart from detecting eavesdropping on plain-text user credentials and HTTP URLs, our system is capable of being used for various other complex forms of traffic eavesdropping and misuse detection. As a proof of concept, we tried to detect HTTP cookie hijack attacks and SSL man-in-the-middle attacks by malicious exit nodes. We elaborate more on these efforts in this section.

5.1 Detection of HTTP session hijacking

Besides snooping on users' traffic, an adversary that has access to unencrypted network data can also mount HTTP session hijacking attacks against users that connect to social networking sites like `facebook.com`. Previously, such sites had no option to encrypt user traffic except while authenticating them. Now, even when using HTTPS, there are various `facebook.com` applications that switch to HTTP and never switch back to HTTPS again, thereby exposing HTTP session cookies to eavesdroppers. In a session hijacking attack, the attacker can steal the session cookie that is included in the HTTP requests of authenticated users and use it to access the user's account. The fact that social networking sites are among the most frequently accessed Web sites Tor [36], combined with the ease of hijacking user sessions using tools like Firesheep [26], makes the possibility of

mounting session hijacking attacks on Tor exit nodes quite attractive for adversaries.

For detecting session cookie hijack attacks, we create several fake `facebook.com` user profiles. In this scheme, the decoy traffic consisted of activity generated by connecting to these `facebook.com` profiles and performing canned activities such as checking messages and status updates. However, we could not create unique `facebook.com` profiles for each of the approximately 900 Tor exit nodes that supported exit for web traffic. We thus created about twenty accounts and repeatedly logged into `facebook.com` by reusing the accounts. We exposed the first account in our list to the first exit node in our list, second account to the second one in our list, and so on till the twentieth account. After logging in, our system checked the various user profile pages and private message folders for new messages. This process exposes the session cookies to the exit node several time for each of the accounts. Thereafter, our system waited for a few minutes and checked the first twenty accounts for changes in profiles such as status update messages or private messages to others users in the contact list of the hijacked profile. Then, we again exposed the first user account in our list to the twenty-first exit node, the second one to the twenty-second and so on. We used a `firefox` browser automation framework, called `iMacros` [29], to perform these periodic logins and canned interactions. We ran our system for about 6 months but did not find any eavesdropping exit nodes sniffing on Facebook cookies.

5.2 SSL man-in-the-middle attack detection

Man-in-the-middle attacks have been observed by some malicious exit nodes [4] that try to intercept and compromise SSL key establishment process and use unverifiable or self-signed certificates. Our system can easily be adapted to detect such man-in-the-middle attacks. To do so, we need to transmit SSL connections via Tor exit node to SSL services whose certificates can otherwise be verified. If in some cases we are unable to verify the server certificate when accessing the server via an exit node, we would conclude that the exit node has possibly manipulated the server to client traffic and might be intercepting the SSL connection establishment. In our setup, we used `curl` to connect to popular HTTPS sites such as popular webmail services and banks and checked whether server certificate could be verified. One could use other tools such as Tor SSL MITM Checker [37] for checking the certificates. This process is repeated for all exit nodes. We found one exit node through which when SSL traffic was exposed, the server certificate verification failed. This node was, however, already blacklisted in the Tor directory services and would not normally be selected when using the default Tor client configuration. In fact, more recently, Winter and Lindskog [55] have demonstrated that there are several exit nodes in Russia which are involved in various kinds of SSL man-in-the-middle attacks. Based on their observations, they claim that these might be operated by the same individual (or group).

6 Discussion and future work

6.1 Detection confidence

Internet traffic crosses multiple network elements until it reaches its final destination. The encrypted communication used in anonymity networks protects the original user traffic from eavesdropping by intermediate network elements, such as routers or wireless access points, until it reaches the boundary of the overlay network. However, the possibility of traffic interception is not eliminated, but is rather shifted to the network path between the exit node and the actual destination. Consequently, the transmitted decoy credentials in our proposed approach might not necessarily be snooped on the exit node of the overlay, but on any other network element toward the destination. This means that in the incidents detected by our system, the decoy credentials could have been intercepted at some other point in the network path between the exit node and the decoy server, and not at the exit node itself.

Although the above possibility can never be ruled out completely, we strongly believe that in all incidents the decoy credentials were indeed intercepted at the involved exit node for the following reasons. The ease of installing and operating a Tor exit node means that not only adversaries can easily set up and operate rogue exit nodes, but also that exit nodes operated by honest individuals may be running on systems that lack the latest software patches, or have poor security configurations. This may enable adversaries to easily compromise them and misuse the hosted Tor exit node. At the same time, most of the network elements beyond a Tor exit node are under the control of ISPs or other organizations that have no incentive to blatantly misuse intercepted user credentials by directly attempting to access the user's accounts. Furthermore, in some of the cases, the adversary connected back to the decoy server from the same exit node involved in the particular eavesdropping incident, raising even more suspicion that the exit node was rogue or has been compromised. In several cases, the involved exit node was inaccessible immediately following the connect-back attempt. In many cases, the eavesdropping adversaries connected back to the decoy server via other exit nodes, so as to hide their true sources. Such typical actions of almost all adversaries shift the suspicion toward Tor relay operators.

Increasing Detection Confidence Using Multiple Decoy Servers. As part of our future work, we plan to use multi-

ple decoy servers scattered in different networks. Thereafter, we could check for eavesdropping and subsequent replay of user traffic, on each of the decoy servers. If eavesdropping is attempted on a traffic going to only a subset of the decoy servers, then it might be due to a malicious network router intercepting the path connecting the exit node to the said decoy servers. If, however, eavesdropping is attempted for the traffic going to all the decoy servers via an exit node, it might have been perpetrated by the said exit node. Furthermore, one may use different sets of user accounts and decoy documents for the different decoy servers. Each of the exit nodes would thus be exposed to multiple sets of decoy user credentials, each one associated with a different decoy server. If, for a given exit node, eavesdropping is detected for one set of decoy user credentials or decoy documents, and not for others, then it might have been on network elements between the exit node and the corresponding decoy server, corresponding to the said set of decoy credentials or documents. However, if eavesdropping is detected for all the decoy user credentials or documents exposed to it, it might likely be involving the exit node, because the network routers in the paths from the exit node to the individual decoy servers are exposed to different decoy user credentials. It seems less likely that a network router on a certain path would know the decoy user credentials that are exposed to routers on other network paths. That said, these measures do not completely rule out the scenarios wherein the adversary happens to eavesdrop on the traffic when the client access only one of the decoy servers and not the other(s) or when the network paths intervening the exit nodes and the decoy servers intersect (and share common network elements).

6.2 Traffic eavesdropping and anonymity degradation

Traffic eavesdropping on anonymous communication systems might not lead to direct degradation of network anonymity. However, inadvertently leaking user information such as login credentials can reveal vital information about the users, such as identity, location, service usage, social contacts, and so on. Specifically for Tor, the *anonymity set* commonly refers to all possible circuits that can be created, or the set of all possible active users of the system [19].

Traffic eavesdropping might help reveal information like the language and content of the messages, the particular dialect of the users, or other peculiarities that might help reducing the size of the anonymity set. For instance, a malicious exit node operator might see traffic carrying user data in Greek. Combined with the knowledge that there are about seven ISP networks in Greece, this information might help reducing the anonymity set significantly. Other clues such as the actual accessed content, the time of access, and the destination of the traffic can as well aid the process of determining a user's identity.

6.3 Lessons learnt

Until recently, there were no adequate research efforts to identify malicious eavesdropping nodes of anonymous communication networks. Popular anonymization networks like Tor are prone to Sybil attacks [23], where an adversary could run a few malicious nodes and attract a large fraction of the traffic [6] to aid *traffic analysis attacks* [3,42]. In such attacks, an adversary, capable of observing network traffic statistics in several different networks, correlates the traffic patterns in them and associates otherwise seemingly unrelated network connections. The process can lead an adversary to the source of an anonymous connection. Such attackers, running malicious nodes (e.g., malicious Tor exit nodes), could eavesdrop on users' traffic to gather private information. In fact, powerful government organizations could be operating nodes with high bandwidth, in several networks, to collect sensitive data and aid traffic analysis attacks.

We have explored ways of identifying such malicious exit nodes by deliberately exposing them with "sensitive appearing" but fake traffic (such as fake usernames and passwords for e-mail accounts), destined to decoy servers under our control. Such eavesdroppers, having access to traffic entering and leaving exit nodes, could also be potential traffic analysis attackers. Therefore, by determining eavesdroppers, one could be potentially identifying attackers who may also launch other types of attacks.

Our prototype system uses decoy traffic for some common TCP/IP services that support plain-text user authentication messages. We have successfully demonstrated that the strategy could be used to identify malicious Tor exit nodes. Based on the modus operandi of the eavesdropping adversaries, and their activities recorded in the logs of the decoy servers, we augmented our system to gather more information about the adversaries and explored possibilities of using our strategy with more advanced kinds of eavesdropping activities, such as HTTP cookie hijacking attacks and SSL man-in-the-middle attacks.

7 Conclusion

Users of various anonymous communication networks and proxying architectures often misconstrue the anonymity guarantees offered by such systems with end-to-end confidentiality. The use of encryption in anonymity networks, like Tor, protects the confidentiality of the user traffic as it is being relayed within the overlay network. This protects the original user traffic against surveillance by local adversaries, as for example in the case where the user is connected through an unsecured public wireless network. Even when encrypted using SSL, users are not safe from man-in-the-middle attacks.

In this paper, we have focused on the problem of detecting malicious eavesdropping nodes of proxying architectures, especially anonymity networks. To tackle this problem, we have presented an approach involving the use of decoy network traffic injection to detect rogue nodes of anonymity networks engaged in traffic eavesdropping. Our approach is based on the injection of bait credentials and decoy documents, through Tor, to decoy services such as IMAP, SMTP, and HTTP, with the aim to entice prospective snoopers to intercept and actually use the bait credentials and documents. The system can detect if a set of credentials has been intercepted, by monitoring for unsolicited connections to the decoy servers and by the alerts generated by the decoy documents containing the beacons, exposed to the exit nodes. We additionally run Honeypots to gather more information of the attacker. Moreover, our system can be easily adapted to detect more advanced traffic interception attacks such as HTTP cookie hijack and SSL man-in-the-middle attack.

Our prototype has been operational for over 30 months. During this period, the system detected eighteen incidents of traffic interception, involving exit nodes across the world. In all cases, the adversary attempted to take advantage of intercepted bait IMAP credentials by logging in on the decoy server, in some cases from the same exit node involved in the eavesdropping incident. Our system continues to run and detect eavesdropping by malicious exit node operators. Details of the latest incidents can be obtained from our Web site [14].

Acknowledgments This work was supported by DARPA and ONR through Contracts DARPA-W011NF-11-1-0140 and ONR-MURI-N00014-07-1-090, respectively. Any opinions, findings, conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect those of the US Government, DARPA, or ONR.

References

1. Anonymizer, Inc. http://www.anonymizer.com/
2. Anonymouse. http://anonymouse.org/
3. Back, A., Möller, U., Stiglic, A.: Traffic analysis attacks and trade-offs in anonymity providing systems. In: Proceedings of the 4th International Workshop on Information Hiding(IHW), pp. 245–257. Springer, London (2001)
4. Known bad relays. https://trac.torproject.org/projects/tor/wiki/doc/badRelays
5. Balsa—An e-mail client for GNOME. http://balsa.gnome.org/
6. Bauer, K., McCoy, D., Grunwald, D., Kohno, T., Sicker, D.: Low-resource routing attacks against tor. In: Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society (WPES), pp. 11–20 (2007)
7. Bauer, K., McCoy, D., Grunwald, D., Sicker, D.: Bitblender: light-weight anonymity for bittorrent. In: Proceedings of the workshop on Applications of private and anonymous communications, AIPACa '08, pp. 1:1–1:8. ACM, New York, NY, USA (2008)doi:10.1145/1461464.1461465
8. Bennett, K., Grothoff, C.: Gnunet: gnu's decentralized anonymous and censorship-resistant P2P framework. http://gnunet.org/
9. Bennett, K., Grothoff, C.: GAP—practical anonymous networking. In: Proceedings of the Privacy Enhancing Technologies Workshop (PET), pp. 141–160 (2003)
10. Bowen, B.M., Hershkop, S., Keromytis, A.D., Stolfo, S.J.: D-cubed. http://sneakers.cs.columbia.edu/ids/RUU/Dcubed/
11. Bowen, B.M., Hershkop, S., Keromytis, A.D., Stolfo, S.J.: Baiting inside attackers using decoy documents. In: Proceedings of the 5th International ICST Conference on Security and Privacy in Communication Networks (SecureComm), pp. 51–70 (2009)
12. Bowen, B.M., Kemerlis, V.P., Prabhu, P., Keromytis, A.D., Stolfo, S.J.: Automating the injection of believable decoys to detect snooping. In: Proceedings of the Third ACM Conference on Wireless Network Security (WiSec), pp. 81–86 (2010)
13. Bowen, B.M., Salem, M.B., Hershkop, S., Keromytis, A.D., Stolfo, S.J.: Designing host and network sensors to mitigate the insider threat. IEEE Secur. Priv. **7**, 22–29 (2009). doi:10.1109/MSP.2009.109
14. Chakravarty, S., Polychronakis, M., Portokalidis, G., Keromytis, A.D.: Details of various eavesdropping incidents. http://dph72nibstejmee4.onion/decoys_via_tor/map.html
15. Charavarty, S., Portokalidis, G., Polychronakis, M., Keromytis, A.D.: Detecting traffic snooping in tor using decoys. In: Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection, pp. 222–241 (2011)
16. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM **24**(2), 84–90 (1981)
17. Claws mail. http://www.claws-mail.org
18. Desaster: kippo ssh honeypot. http://code.google.com/p/kippo
19. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Proceedings of the 2nd International Conference on Privacy Enhancing Technologies. PET'02, pp. 54–68. Springer, Berlin (2003)
20. Dingledine, R., Mathewson, N.: Tor path specification. https://gitweb.torproject.org/torspec.git?a=blob_plain;hb=HEAD;f=path-spec.txt
21. Dingledine, R., Mathewson, N., Syverson, P.: Onion Routing. http://www.onion-router.net/
22. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: Proceedings of the 13th USENIX Security Symposium, pp. 303–319 (2004)
23. Douceur, J.R.: The sybil attack. In: Proceedings of International Workshop on Peer-to-Peer Systems (2001)
24. Stenberg, D.: kippo curl. http://curl.haxx.se
25. Evolution. http://projects.gnome.org/evolution
26. Firesheep. http://codebutler.com/firesheep
27. The Honeynet Project. http://www.honeynet.org/
28. I2P Anonymous Network. http://www.i2p2.de/
29. iOpus™: iMacros©. http://www.iopus.com/imacros/
30. Isdal, T., Piatek, M., Krishnamurthy, A., Anderson, T.: Privacy-preserving P2P data sharing with oneswarm. In: Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM), pp. 111–122 (2010)
31. JAP. http://anon.inf.tu-dresden.de/
32. Kmail—mail client. http://kde.org/applications/internet/kmail
33. McCanne, S., Leres, C., Jacobson, V.: Tcpdump and libpcap. http://www.tcpdump.org/
34. Mccoy, D., Bauer, K., Grunwald, D., Kohno, T., Sicker, D.: Shining light in dark places: understanding the tor network. In: Proceedings of the 8th International Symposium on Privacy Enhancing Technologies (PETS), pp. 63–76 (2008)
35. Meyers, J.: IMAP4 ACL extension. http://www.ietf.org/rfc/rfc2086.txt
36. Mulazzani, M., Huber, M., Weippl, E.R.: Tor HTTP usage and information leakage. In: Proceedings of the IFIP Conference on

Communications and Multimedia Security (CMS), pp. 245–255 (2010)
37. Palfrader, P.: Tor SSL MITM check. http://svn.noreply.org/svn/weaselutils/trunk/tor-exit-ssl-check
38. Pound, C.: Chris Pound's language machines. http://www.ruf.rice.edu/~pound/
39. Pound, C.: Language confluxer. http://www.ruf.rice.edu/~pound/new-lc/
40. Pound, C.: Prop. http://www.ruf.rice.edu/~pound/prop
41. Provos, N.: A virtual honeypot framework. In: Proceedings of the 13th USENIX Security Symposium, pp. 1–14 (2004)
42. Raymond, J.F.: Traffic analysis: protocols, attacks, design issues, and open problems. In: Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability, pp. 10–29. Springer, LNCS 2009 (2000)
43. Reed, M.G., Syverson, P.F., Goldschlag, D.M.: Anonymous connections and onion routing. IEEE J. Sel. Areas Commun. **16**, 482–494 (1998)
44. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. ACM Trans. Inf. Syst. Secur. **1**, 66–92 (1998)
45. Services, O.U.C.: The university of oxford text archive. http://ota.ahds.ac.uk/
46. Spitzner, L.: Honeytokens: the other honeypot. http://www.symantec.com/connect/articles/honeytokens-other-honeypot
47. Spitzner, L.: Honeypots: catching the insider threat. In: Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC) (2003)
48. Stoll, C.: Stalking the wily hacker. Commun. ACM **31**(5), 484–497 (1988)
49. Stoll, C.: The cuckoo's egg: tracking a spy through the maze of computer espionage. Doubleday, New York (1989)
50. Sylpheed-lightweight and user-friendly e-mail client. http://sylpheed.sraoss.jp/en
51. Furry, T.: TOR exit-node doing MITM attacks. http://www.teamfurry.com/wordpress/2007/11/20/tor-exit-node-doing-mitm-attacks/
52. Tor metrics portal. http://metrics.torproject.org/
53. Tor metrics portal: number of users. http://metrics.torproject.org/users.html
54. Ts'o, T.: Password generator. http://sourceforge.net/projects/pwgen/
55. Winter, P., Lindskog, S.: Spoiled onions: exposing malicious tor exit relays. Technical Report, Karlstad University (2014). URL http://veri.nymity.ch/spoiled_onions/techreport.pdf
56. Yuill, J., Zappe, M., Denning, D., Feer, F.: Honeyfiles: deceptive files for intrusion detection. In: Proceedings of the 2nd IEEE Workshop on Information Assurance (WIA), pp. 116–122 (2004)