Detecting Traffic Snooping in Anonymity Networks Using Decoys

Sambuddho Chakravarty, Georgios Portokalidis, Michalis Polychronakis, and Angelos D. Keromytis

Columbia University, NY, USA
{sc2516,porto,mikepo,angelos}@cs.columbia.edu

Abstract. Anonymous communication networks like Tor partially protect the confidentiality of their users' traffic by encrypting all intra-overlay communication. However, when the relayed traffic reaches the boundaries of the overlay network towards its actual destination, the original user traffic is inevitably exposed. At this point, unless end-to-end encryption is used, sensitive user data can be snooped by a malicious or compromised exit node, or by any other rogue network entity on the path towards the actual destination.
We explore the use of decoy traffic for the detection of traffic interception on anonymous proxying systems. Our approach is based on the injection of traffic that exposes bait credentials for decoy services that require user authentication. Our aim is to entice prospective eavesdroppers to access decoy accounts on servers under our control using the intercepted credentials. We have deployed our prototype implementation in the Tor network using decoy IMAP and SMTP servers. During the course of six months, our system detected eight cases of traffic interception that involved eight different Tor exit nodes. We provide a detailed analysis of the detected incidents, discuss potential improvements to our system, and outline how our approach can be extended for the detection of HTTP session hijacking attacks.

1 Introduction

Internet users often place trust in various systems that are not directly under their control. With the emergence of cloud computing, and the continuously increasing number of services migrating to the cloud, it is more so today than ever. Anonymity and privacy-preserving systems like Tor [13], Anonymizer [1], and many others [23,17,2,16,6,11] are such systems. They operate by routing user traffic through a single or multiple proxies to achieve a twofold goal. First, they preserve user anonymity, and second, they enable users to access services and content which might otherwise be restricted to them. For example, anonymity networks enable users to avoid being tracked by governments and Internet service provides (ISPs) when accessing restricted content [3,30].

Users of such anonymous communication systems are able to conceal information such as their IP address by the provider of the end-service. In exchange,

they place their trust in components of the communication system they are using. In all cases, user data are at some point (for instance, before being relayed to the end-service) available in their original form. Even if encryption is utilized by the system internally, end-to-end encryption is imperative to ensure the confidentiality of user communications. In practice, *users frequently confuse the anonymity and privacy guarantees offered by these systems with data confidentiality, and use them without employing end-to-end encryption, revealing their data to the proxies relaying them.*

In this paper, we explore the use of decoy traffic to detect eavesdropping in proxying architectures, and in particular anonymous communication systems. We introduce decoy credentials for various services, like SMTP, in the Tor anonymity network, and use them to detect snooping exit nodes. The use of fake information or *honeytokens* [25] is not new. Decoy information has been previously used to to detect eavesdropping on unprotected wireless networks [8], and warn of insider threats [7]. The idea behind their use is that eavesdroppers will probably try to use the collected information in some way. By injecting login credentials for services that we control, we are able to detect the use of a particular decoy, and trace it back to the Tor exit node that it was used with.

Tor is one of the most popular anonymity networks, based on onion routing [12] and Chaum's mixes [10]. Tor clients form virtual circuits consisting of two or more Tor nodes, which relay client traffic to the intended server. Their data is encrypted multiple times before being transmitting over the Tor network, so that the original data are available only at the exit node (that is, the last node in the circuit). As such, unless end-to-end encryption between a client and a service is used, the confidentiality of the data can be potentially undermined. For instance, data could be eavesdropped by a malicious or compromised exit node, or even by the ISP of the exit node. In fact, all proxying architectures face the same threat, unless end-to-end encryption is used. We evaluate our approach using the Tor network exactly because Tor has a considerable user base, and because its users frequently fail to use end-to-end encryption.

McCoy et al. [19] also investigated eavesdropping in the Tor network. However, their methodology significantly differs from ours. In their setup, a client performs a connection to a network domain under their control, and eavesdropping is detected by monitoring the DNS server of that domain for reverse lookup queries. The authors reported that they detected eavesdropping by one Tor exit node, but the methodology does not offer conclusive results, nor can it accurately detect the exit node responsible for the information leak.

Our prototype uses multiple decoy credentials for an IMAP and an SMTP service under our control. We use these decoys to connect to these services through Tor, using every publicly available exit node. The decoys are transmitted in plain-text, and each decoy is only sent through a single exit node, allowing us to pair the use of a particular decoy with an exit node. It has been operational for about six months, and has so far detected eight cases of eavesdropping by public Tor exit nodes.

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

– We present a generic method for the detection of traffic interception in anonymity networks and proxy servers in general, based on the transmission of decoy user credentials.
– We deployed a prototype system for the Tor anonymity network, which detected *eight cases* where decoy credentials were used by a third-party to log in to decoy servers under our control.
– We discuss how our method can be extended to detect HTTP session hijacking attacks, which can be used to take over active user sessions on websites where encryption is not used throughout a session [14]. Note that such attacks are possible even when a user transmits his credentials using encryption (for example, over HTTPS), but accesses a website in plain-text thereafter. As is the case with websites like Facebook and Twitter.

The next section provides some background information on the Tor anonymity network, and presents the threat model we are considering. Section 3 describes the design and implementation of our decoy transmission and eavesdropping detection engine. We present the results obtained by deploying our prototype in Section 4. In Section 5, we discuss limitations, and possible extensions to our system, including the detection of HTTP session hijacking. Finally, related work is discussed in Section 6, and conclusions are in Section 7.

2 Background

In this section we briefly describe the architecture of the Tor anonymity network, and present the threat model assumed in this work.

2.1 Tor Anonymity Network

Tor [13] is one of the most widely used low latency anonymity networks, with an estimated user base of more than 150,000 users as of January 2011 [5]. Tor aims to protect the anonymity of Internet users by relaying user-generated TCP streams through a network of overlay nodes run by volunteers. Tor can be used for both *initiator* and *responder* anonymity. Initiator anonymity hides the true identity (IP address) of user-initiated connections from the actual destination, while the identity of network servers can also be kept secret from their clients through the use of *hidden services*.

The Tor overlay network consists of hundreds of proxies known as *onion routers*, which are mostly operated by volunteers around the world. User traffic is relayed through *circuits*, which are formed by persistent 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 one as the *exit node*. A Tor client uses the public keys of the onion routers on the circuit to encrypt transmitted messages in multiple layers of encryptions, 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.

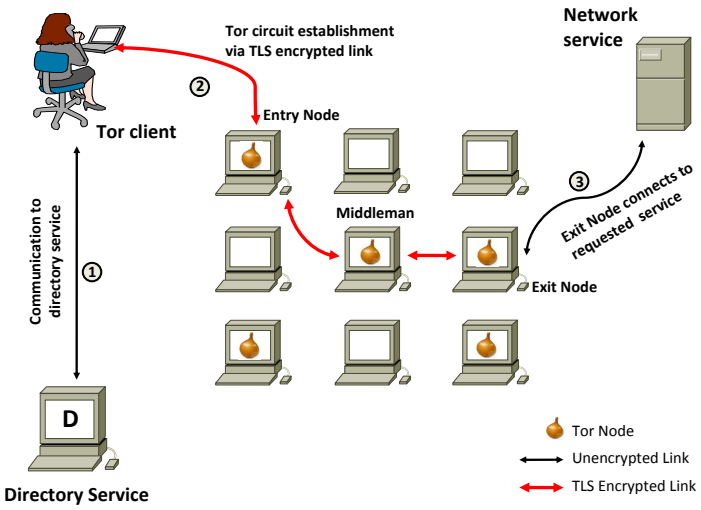

Fig. 1. Basic steps for communicating through Tor. The client obtains a list of the available Tor relays from a directory service ①, establishes a circuit using multiple Tor nodes ②, and then starts forwarding its traffic through the newly created circuit ③.

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.

Figure 1 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.

2.2 Threat Model

Exit nodes act as proxies between the user and the actual destination. This places them in a powerful position that allows malicious exit node operators to take advantage of their access to the user's original network traffic. Consequently, the trust that the users place on an anonymous communication service like Tor can be affected by misbehaving or compromised overlay nodes. A rogue exit node can capture all the incoming and outgoing user traffic between the exit node

and the actual destinations. We expect the attacker to sift through the captured user traffic and extract user credentials from clear-text application protocols. This can be easily achieved using custom tools built on top of `libpcap` [18], or through the use of existing tools like `dsniff` [24]. Of course, the attacker might be eavesdropping for a particular kind of private information, such as the content of email messages [4], which can then be misused in other, non-obviously detectable ways.

Credentials such as user names and passwords or sessions cookies can be reused by the attacker on the same destination server. These might allow him to take over the user's account for that service or hijack an ongoing session. Note that the attacker's connections using the stolen credentials can be launched either from the same host that runs the malicious Tor node, or any other host on the Internet.

Besides unencrypted traffic, even properly encrypted user connections such as HTTPS sessions to banking or webmail sites can be compromised by malicious exit nodes. For example, an attacker can mount a man-in-the-middle attack and intercept the traffic of SSL connections [21]. Attacks of this kind can be easily detected [21,29], an thus are out of the scope of this work.

3 System Architecture

In this section, we present the overall architecture of our traffic interception detection system. 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. Finally, we discuss some interesting implementation issues that we faced during the development of our prototype system.

3.1 Approach

In general, network traffic eavesdropping is a passive operation without any directly observable effects. However, the fact that some traffic has been intercepted can be implied through potential uses of the intercepted data that have detectable corollaries. For example, the eavesdropper can steal user credentials for services that do not use application-layer encryption, such as user names and passwords for websites with poor user authentication implementations, or for servers that use clear-text sign-in protocols, such as FTP or IMAP. A later attempt by the eavesdropper to access the user's account is an observable event that can be detected by the operator of the respective service.

Our approach is based on enticing a prospective snooper to use intercepted decoy credentials for accessing a service under our control. The proposed system transmits decoy credentials through network paths on which there is a possibility of traffic eavesdropping. Each set of credentials is unique, has never been used before, and is being transmitted solely through a specific network path. All subsequent unsolicited access to one of the accounts on the decoy server are

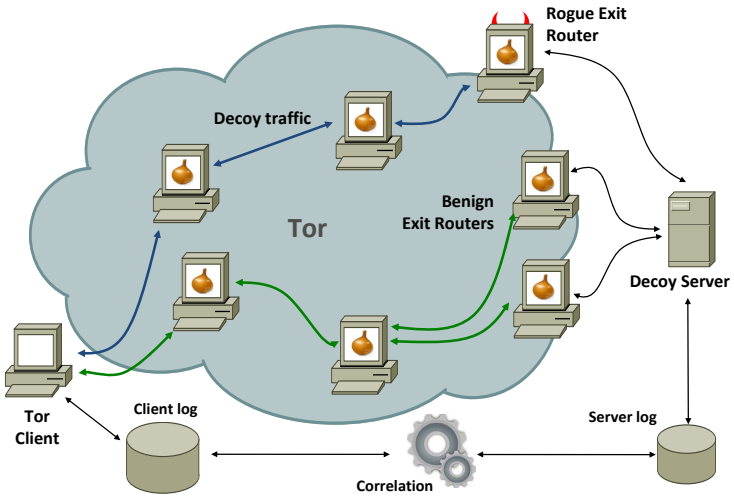

Fig. 2. Overall architecture of the proposed traffic interception detection system when applied on the Tor network.

clear indications that the credentials tied to that accounts have been intercepted during their transmission.

Figure 2 illustrates the overall design of our system when applied on the Tor anonymity network. A client under our control periodically connects through Tor to a decoy server, which uses a clear-text application-level protocol requiring password authentication. As a result, the user name and password used in each session are exposed to the exit node of the Tor circuit (and any other network entity between the exit node and the decoy server).

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

1. The client connects and authenticates to the decoy server through the Tor network. A new connection using a different set of credentials is made through *all* available exit nodes by explicitly specifying the exit node of each Tor circuit.
2. The decoy server keeps a detailed record for each session, including the user name and password used, the IP address of the connection initiator, and the login time.
3. After a successfully completed session on the decoy server, the system attempts to correlate it with a recently completed client session. If no matching client session is found, then an unsolicited connection using stolen decoy credentials has been identified.

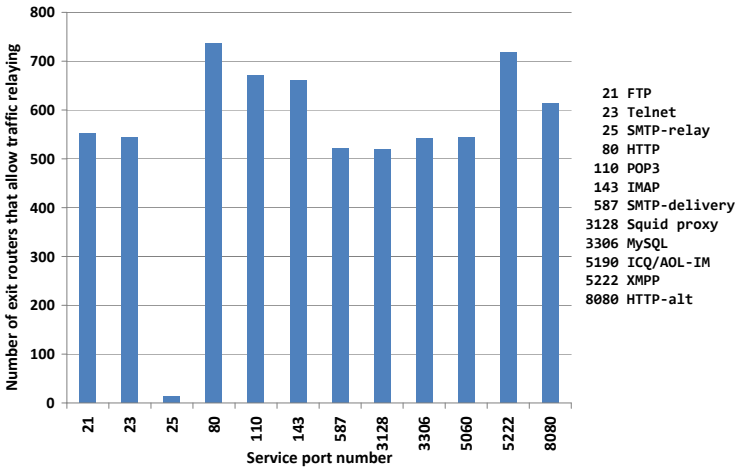

Fig. 3. Number of Tor exit routers that allow traffic relaying through different TCP port numbers, for services that support clear-text protocols.

Each unique pair of user name and password is tied to a particular exit node and is transmitted only through Tor circuits terminating at that node. Thus, the exit node involved in a particular eavesdropping incident is known based on the given set of credentials used in the unsolicited session seen by the decoy server. At the same time, the server is aware of the IP address of the connection initiator, which, as discussed in Section 4, may belong either to the rogue exit node itself, or to a third-party host on the Internet.

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 be enticing for potential eavesdroppers.

Tor exit nodes are usually 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*. 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 to the results obtained by McCoy et al. [19], widely used protocols for applications like web browsing, email retrieval, and instant messaging are allowed by the majority of exit nodes. Among the services that support user authentication through unencrypted protocols, IMAP (port 143) and SMTP delivery (port 587) are allowed by the exit policies of a significant number of exit nodes (661 and 522 nodes, respectively). 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.

Credentials for accessing user messages that may contain sensitive private information, or for sending emails 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. However, our technique is not restricted to these two services, and can easily be extended to include bait traffic for various other unencrypted TCP-based services like FTP, Telnet, and instant messaging. In Section 5 we also discuss how our technique can be extended to detect the interception of user login credentials and cookies for various web services.

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. The client is hosted on a server equipped with an Intel Xeon CPU running Ubuntu Server Linux v8.04.

Every day, for each service, the client creates one connection to the corresponding decoy server through each and every Tor exit node that supports traffic relaying for that service. This is achieved by establishing a new Tor circuit for each connection, and forcing 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 tied to the particular combination of exit node and decoy server. 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. After the client has successfully signed in, it generates some randomly selected activity such as browsing through some folders in case of IMAP, or sending a fake email message in case of SMTP, and then signs out.

For the decoy services we use Courier IMAP v4.6.0 and Postfix v2.7.0 running on a different host. Under normal conditions, each decoy server should receive one connection from each unique account per day. If an unsolicited successful connection using some of the previously transmitted decoy credentials is observed, then this connection is labelled as illegitimate. Illegitimate connections are identified by correlating the connections generated by our client with all the connections received by the server, based on the logs kept 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 used and the start and end times of the connection recorded by both the client and the server, against the recently completed connections. 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.

Implementation Details.

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

Time Synchronization. Accurate time synchronization between the client and the decoy server(s) is crucial for ensuring the 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.

Amount and Quality of Decoy Traffic. We deliberately chose to generate a conservatively small number of decoy connections instead of sending an increased amount of decoy network traffic. On one hand, the probability that some of the transmitted decoy credentials will be snooped increases with the number and frequency of the generated decoy connections, e.g., in case of intermittent traffic interception or opportunistic eavesdroppers. At the same time, as the amount of decoy traffic increases, it can potentially become more distinguishable from the production traffic. Although keeping the number of decoy connections to one per day for each combination of exit node and decoy service may not provide the higher possible exposure of the bait credentials to prospective eavesdroppers, it makes the identification of the decoy traffic much harder.

The believability of the decoy traffic [8] is another 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 email 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 randomly vary the connection times and activity in each session, we use realistically looking folder structures for the IMAP accounts, and

send legitimately looking email messages that are randomly selected from a pool of existing messages.

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 misclassification 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 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 or SMTP client.

4 Deployment Results

Our prototype implementation has been continuously operational in the Tor anonymity network since August 2010. During the course of six months, our system has detected eight traffic interception incidents. In this section, we give a detailed description of each incident and an analysis of the consequent activity on the decoy server.

The observed eavesdropping incidents were related to eight 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. Detailed information about the detected incidents is presented in Table 1.

The first four incidents occurred within a short timespan of three days, and involved four different exit nodes in the US, Hong Kong, UK, and The Netherlands. The connect-back attempts on the decoy server had a common pattern, and in all four cases they originated from the same IP address of the exit node on which the corresponding credentials had been exposed. Another similarity among these incidents is related to time difference between the latest exposure of the decoy credentials in the network and the corresponding connect-back to the decoy server. Figure 4 presents this time difference for all eight incidents. The first four incidents had a quite similar connect-back delay of a few hours, which is significantly shorter compared to the rest of the incidents. Based on the above facts, we speculate that the first four eavesdropping cases were coordinated by the same person or group, who probably used the same tools or methodology in each case.

The fifth incident occurred about three weeks after the previous group of incidents. The decoy user name and password were exposed through an exit router in South Korea, and a connection to the decoy server was attempted from a *different* exit router in the US—an indication that the adversary probably used Tor to hide the real origin of the connection. The sixth incident almost coincided

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

Table 1. Observed traffic interception incidents during a six-month period. In all cases, the eavesdropper connected to our decoy IMAP server using a set of intercepted decoy credentials.

with the fifth one, and involved an exit router in Hong Kong. After more than ten hours, the decoy IMAP server received six connections from a different IP address belonging to a Chinese ISP.

In the seventh eavesdropping case, the decoy user credentials were exposed through an exit router located in India. The credentials were then reused in five connections originating from five different IP addresses within the same subnet of an ISP in Canada. Interestingly, the exit router was not accessible when we discovered the eavesdropping attempt. An analysis of the network traffic captured on the decoy server revealed that in each session, there were multiple accesses to default mail folders such as `INBOX`, `INBOX.Sent`, and `INBOX.Template`, although some of them (e.g., `INBOX.Template`) didn't exist in the decoy account. This is an indication that the attacker probably used an email client that automatically attempts to browse through some standard folders.

The latest incident occurred in the first week of January 2011 and involved an exit node in Germany. Five unsolicited connections were received by the decoy server from a host located in Ecuador. In all cases, upon successfully

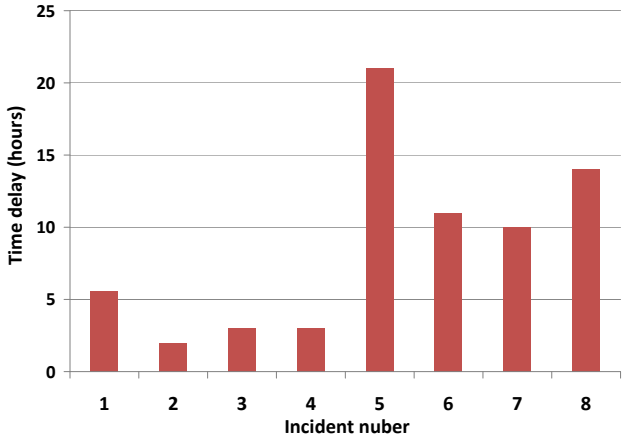

Fig. 4. Time difference between the exposure of the decoy credentials and the first connect-back attempt on the decoy server.

authenticating on the decoy server, the mail client of the adversary issued an IMAP `STARTTLS` command, attempting to switch to an SSL connection.

The map in Figure 5 gives an overall view of the 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.

5 Discussion and Future work

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 towards 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

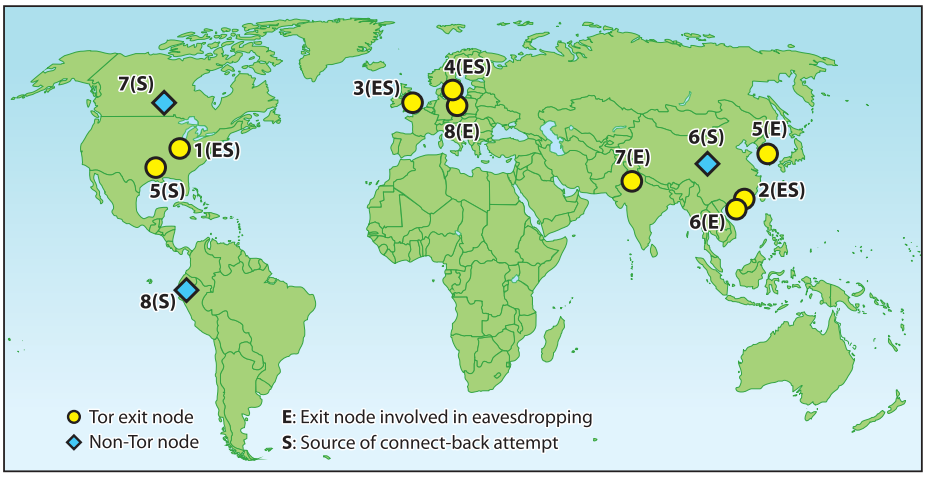

Fig. 5. Locations of the Tor exit nodes involved in the observed traffic interception incidents, and the non-Tor hosts that connected back to the decoy servers. Numbers refer to the corresponding incidents listed in Table 1.

involved exit node for the following reasons. The ease of installing and operating a Tor exit node means that 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 half 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 is rogue or has been compromised.

In our future work, we plan to use multiple replicas of the decoy servers scattered in different networks around the world, and associate different sets of credentials with each one. This can further increase the detection confidence for incidents involving the same exit node, but different replicas of the same server.

Decoy Traffic Credibility.

Another aspect of our system that can be improved is the credibility of the generated decoy traffic. For instance, regarding the SMTP traffic, we plan to increase the number and diversity of the innocuous email messages that we currently use, and also create new variations based on message templates. Some of the messages could also contain "bait" documents [9] that would ping back to our system in case someone opened them. We can also use some of the techniques described

by Bowen et al. [8] to generate even more realistic decoy traffic. For example, we can capture network traces of protocol interactions using various real IMAP clients and servers, sanitize and modify them by inserting bait information, and replay them as part of the decoy traffic.

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 are logged in on social networking sites like Facebook or Twitter. Although e-commerce and banking sites generally use HTTPS throughout the whole user session, many popular sites by default use HTTPS only for user authentication, and then switch to plain HTTP. 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 websites through the Tor network [20], combined with the ease of hijacking user sessions using tools like Firesheep [14], makes the possibility of mounting session hijacking attacks on Tor exit nodes quite attractive for adversaries.

In our future work, we plan to extend our system to detect HTTP session hijacking attacks through the use of decoy accounts on popular social networking websites. In this scheme, the decoy traffic will consist of generated random activity using decoy accounts on websites like Facebook. This activity can include actions such as viewing pictures, browsing through friends' posts, or sending fake messages. Instead of decoy credentials, our aim in this case would be to entice a potential adversary to intercept the session cookie used in the decoy HTTP requests and hijack the fake user's session. The hijacking event can be detected by closely monitoring all information contained in the decoy account for potential changes that would indicate that someone has gained unauthorized access. For instance, an attacker might use a hijacked Facebook account to post links to malicious code or send spam messages to the friends of the user.

6 Related work

Our work is closely related to research efforts that involve the exposure of enticing decoy information or resources to potential adversaries, with the aim to observe who and how attempts to use it. One of the first uses of decoy information for enabling the observation of real malicious activity has been documented by Clifford Stoll [27]. In his book, *The Cuckoo's Egg* [28], 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.

The use of decoy computers with the aim to lure prospective intruders and monitor their actions is nowadays a popular approach among security administrators and researchers. These systems, widely known as *honeypots* [22,26], have no production value other than being compromised, and subsequently track the actions of the attacker. Honeypots have been extensively used for modeling, logging, and analyzing attacks originating from sources external to an organization [15,31], as well as internal attacks launched from within its perimeter [9].

Similarly to honeypots, *honeytokens* [25] are pieces of information with no real production value other than being intercepted by an adversary. After their release, any subsequent use of that information can clearly indicate unauthorized access. The decoy credentials used in our approach can thus be viewed as particular instance of honeytokens. Bowen et al. [7] proposed the use of decoy documents to detect misbehaving entities within the perimeter of an organization. The decoy documents contain embedded "beacons," such as scripts or macros, 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 has been leaked, the embedded beacon will connect to an external host and notify its author whenever the document is accessed.

There has been little effort in detecting misbehaving overlay nodes in anonymity networks. In a work most closely related to ours, McCoy et. al [19] attempt 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` [18], are by default configured to resolve the IP addresses of the captured packets to their respective DNS names. The proposed system transmits, 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 attempts to resolve the IP address of a probe packet, it will issue a DNS request to the authoritative DNS server, which is also under the control of the system's operator. Thus, any observed unsolicited requests to this DNS server are an indication that probe packets have been intercepted by some packet capturing program, and can 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 other case the eavesdropper can trivially disable this functionality, rendering the above technique ineffective.

7 Conclusion

Anonymous communication networks and proxying architectures offer an important service for users that want to protect their anonymity on the Internet. Through the use of encryption, anonymity networks like Tor also protect the confidentiality of the user traffic as it is being relayed across 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. However, since these systems by design do not provide end-to-

end encryption, when the traffic reaches the final node of the overlay network, it is being exposed to potential eavesdroppers.

In this paper, we apply the concept 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 for fake services such as IMAP and SMTP, with the aim to entice prospective snoopers to intercept and actually use the bait credentials. The system can then detect that a set of credentials has been intercepted, by monitoring for unsolicited connections to the decoy servers that use a set of previously exposed bait credentials.

We have deployed our prototype implementation in the Tor network, where it has been operational for more than six months. During this period, the system detected eight incidents of traffic interception, involving eight different 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 many cases from the same exit node involved in the eavesdropping incident.

As part of our future work, we plan to use more decoy services and increase the believability and diversity of our bait traffic, vary the location of the decoy servers, and use multiple replicas of each service in different networks. We also plan to extend our system to detect HTTP session hijacking attacks against popular social networking websites.

References

1. Anonymizer, Inc. http://www.anonymizer.com/
2. Anonymouse. http://anonymouse.org/
3. Inside Net Neutrality: Is your ISP filtering content? http://www.macworld.com/article/132075/2008/02/netneutrality1.html
4. Rogue Nodes Turn Tor Anonymizer Into Eavesdropper's Paradise, http://www.wired.com/politics/security/news/2007/09/embassy_hacks
5. Tor Metrics Portal. http://metrics.torproject.org/
6. Bennett, K., Grothoff, C.: GAP - practical anonymous networking. In: Proceedings of the Privacy Enhancing Technologies Workshop (PET). pp. 141–160 (2003)
7. 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 (September 2009)
8. 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)
9. 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 Security and Privacy 7, 22–29 (2009)
10. Chaum, D.L.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM 24(2), 84–90 (February 1981)
11. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: A Type III Anonymous Remailer. http://mixminion.net/
12. Dingledine, R., Mathewson, N., Syverson, P.: Onion Routing. http://www.onion-router.net/

13. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Proceedings of the 13th USENIX Security Symposium). pp. 303–319 (August 2004)
14. Firesheep. http://codebutler.com/firesheep
15. The Honeynet Project. http://www.honeynet.org/
16. 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)
17. JAP. http://anon.inf.tu-dresden.de/
18. McCanne, S., Leres, C., Jacobson, V.: Tcpdump and Libpcap, http://www.tcpdump.org/
19. 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)
20. 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)
21. Nikiforakis, N., Younan, Y., Joosen, W.: Hproxy: client-side detection of ssl stripping attacks. In: Proceedings of the 7th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA). pp. 200–218 (2010)
22. Provos, N.: A virtual honeypot framework. In: Proceedings of the 13th USENIX Security Symposium. pp. 1–14 (Aug 2004)
23. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. ACM Trans. Inf. Syst. Secur. 1, 66–92 (November 1998)
24. Song, D.: dsniff, http://www.monkey.org/~dugsong/dsniff/
25. Spitzner, L.: Honeytokens: The Other Honeypot. http://www.symantec.com/connect/articles/honeytokens-other-honeypot
26. Spitzner, L.: Honeypots: Catching the insider threat. In: Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC) (2003)
27. Stoll, C.: Stalking the wily hacker. Communications of the ACM 31(5), 484–497 (1988)
28. Stoll, C.: The cuckoo's egg: tracking a spy through the maze of computer espionage. Doubleday, New York, NY, USA (1989)
29. Team Furry: TOR exit-node doing MITM attacks, http://www.teamfurry.com/wordpress/2007/11/20/tor-exit-node-doing-mitm-attacks/
30. Weaver, N., Sommer, R., Paxson, V.: Detecting forged tcp reset packets. In: Proceedings of the 16th Network and Distributed System Security Symposium (NDSS) (2009)
31. 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)