ISPY: A System to Identify Misbehaving Routers in the Internet Using Secret Tester Nodes

Francis Browne
Dept. of Computer Science
Yale University
New Haven, CT 06520
Francis.Browne@Yale.edu

Abstract

Recent events have sparked a renewed interest in securing our nation’s infrastructure. A key component of this infrastructure is the Internet. Today, the Internet plays an important role in many sectors of the economy, from banking and finance to intelligence and defense. Initially constructed for scientists sharing unclassified research, the Internet was not designed with security in mind. However, with the globalization and commercialization of the Internet, it can no longer be assumed that all agents participating in the Internet are trustworthy. In considering potential attacks on the Internet, one of the most attractive targets is the router. Routers are responsible for properly forwarding all data that flows through the Internet. If a malicious agent can gain control of a router, he can cause significant damage by disrupting data flow and communication. In order to prevent this potential for interference, the objective of this project is to study methods of identifying misbehaving routers in a wired network such as the Internet. We present a system called ISPY that is capable of detecting routers that maliciously drop packets. ISPY involves two components: a set of secret tester nodes scattered throughout the Internet and a distribution center that responds to queries for the address of a single secret tester node. In this paper we also analyze the ISPY architecture to determine the optimal number and placement of secret tester nodes. We argue that with ISPY, it is significantly more difficult for a misbehaving router to evade detection. Additionally, we show that ISPY is not burdened with the prohibitive state requirements of previous approaches.
1. Introduction

Recently, there has been a renewed interest in securing our nation’s infrastructure, and a key component of the communication infrastructure is the Internet. Increasingly, the Internet supports broad sectors of the economy, from banking and finance to intelligence and defense. Initially conceived as a means for scientists to share unclassified research, the Internet was not designed with security in mind. One could safely presume that the scientists who constructed the Internet could be trusted not to abuse the network. However, with the globalization and commercialization of the Internet, it can no longer be assumed that all agents participating in the Internet are trustworthy.

In considering potential attacks on the Internet, one of the most attractive targets is the router. Routers are responsible for properly forwarding all data that flows through the Internet. Therefore, if malicious agent can gain control of a router, he can cause significant damage by disrupting data flow and communication. In order to prevent this potential for interference, the objective of this project is to study methods of identifying misbehaving routers in a wired network such as the Internet. For this project, we restrict our analysis to the data plane and we assume that the goal of a malicious router is to restrict the flow of traffic in the Internet. A malicious router can achieve this goal in one of three ways: it can either delay, drop or truncate packets. Also, for the purpose of this project, we employ the idea of trust boundaries. That is, we assume that an autonomous system can trust that none of its internal routers are misbehaving, so we are only concerned with interactions between gateway routers using the border gateway protocol.

Previous approaches to dealing with router misbehavior have primarily focused on protecting routers within a single autonomous system. Perlman suggested the use of hop-by-hop
acknowledgements to expose misbehavior [1]. However, this adds considerable overhead for each router along the path and significantly increases traffic in the Internet with all of the extra acknowledgments that must be sent. Another method proposed involves route tracing, but an intelligently misbehaving router would be able to identify a test packet and thereby avoid detection [2]. A more sophisticated approach for identifying router misbehavior was proposed by Bradley et al. [2]. While his approach is attractive for monitoring misbehavior within a small autonomous system, the state requirements for each router are prohibitive in a larger network setting such as the Internet.

In this paper we present a solution, ISPY, which is capable of identifying router misbehavior in Internet. ISPY involves two components: a set of secret tester nodes scattered throughout the Internet and a distribution center that responds to queries for the address of a single secret tester node. When a router wants to test a neighbor for misbehavior, it contacts the distribution center to get the address of a secret tester node. The router then sends the secret tester node a series of packets and records how many packets were successfully received. If the success rate is below some threshold, the router’s neighbor can be marked as misbehaving. With the proper security mechanisms, we believe ISPY represents an improvement over previous approaches. With ISPY, it is significantly more difficult for a misbehaving router to evade detection. Furthermore, ISPY is not burdened with the prohibitive state requirements of previous approaches.

The remainder of our paper is structured as follows. In Section 2 we provide a background on the structure of the Internet and discuss related work. We describe the ISPY system and the security mechanisms required in Section 3. In Section 4 we present an analysis of ISPY. Finally, we conclude the paper in Section 5 and suggest future work.
2. Background and Previous Work

In order to develop a better understanding of the problem, it is first necessary to examine the nature and structure of the Internet. The Internet connects Autonomous Systems (ASes), such as Internet Service Providers (ISPs), companies and universities. ASes in the Internet are identified by a unique 16-bit number, thus allowing for a total of 65,536 possible ASes. However, current estimates suggest that there are significantly fewer identifications in use. Routing in an autonomous system is governed by intradomain routing protocols such as static routing, OSPF, IS-IS and RIP. However, routing between ASes is controlled by the Border Gateway Protocol (BGP).

Interconnection between ASes can be generally classified into peering relationships and transit relationships. A peering relationship occurs when two ASes reciprocally provide connectivity to each others’ transit customers. However, a peering relationship in the strictest sense is non-transitive [3]. This means that if AS \( X \) peers with AS \( Y \), and AS \( Z \) also peers with AS \( Y \), traffic from \( X \) cannot reach destination \( Z \) by passing through \( Y \). With a transit relationship, on the other hand, an AS provides connectivity to all destinations in its routing table to another AS. Thus if \( X \) peers with \( Y \) and \( Z \) is a transit customer of \( Y \), then \( X \) can reach \( Z \) through \( Y \) and vice versa. Typically, ASes enter into peering relationships when it is mutually beneficial, whereas ASes charge a fee for providing transit service [3].

For the purposes of this project, we focus solely on routers misbehaving at the AS level. There are numerous reasons why these routers might be delaying, dropping or truncating packets. As mentioned in the introduction, because routers control traffic flow in the Internet, they are an attractive target for attackers looking to disrupt global communication. More subtly, however, even ASes that are not under the control of an attacker may actually have an incentive
to drop the packets of a transit customer or a peer. Consider the case of a tier-1 ISP providing transit service to a number of ASes. If the ISP is a profit maximizing firm, it might be in its best interest to sell service to as many customers as possible without regard to whether or not it has the resources to support all of the bandwidth required. A similar argument can be made for peering relationships. The more an AS pairs with other ASes, the less it must pay for transit service. However, as a result of the peering relationships an AS establishes, it may not actually have sufficient hardware (bandwidth, buffer space, etc.) to support all of the traffic. Thus, in both cases an AS might provide a quality of service that is lower than what was initially negotiated.

Many techniques for identifying misbehaving routers have been proposed. Perlman suggested the use of hop-by-hop acknowledgements to expose misbehavior [1]. When a source sends a packet to a destination, the source receives an acknowledgement of the packet from the destination. In addition, the source receives an acknowledgement from each router on the path from the source to the destination. Thus, when a source does not get an acknowledgement from the destination, the bad router must either be the first router along the path which did not send back an acknowledgement or the previous router along the path. However, this adds considerable overhead for each router along the path and significantly increases traffic in the Internet with all of the extra acknowledgments that must be sent.

Another method proposed involves route tracing [2]. Route tracing allows for the selective identification and testing of the routers along a path from a source to a destination. This takes care of the problem of overhead introduced by the previous approach. However, an intelligently misbehaving router would be able to identify a test packet and thereby avoid detection.
A more sophisticated approach for identifying router misbehavior was proposed by Bradley et al. A distributed network monitoring approach, the WATCHERS protocol was designed to protect routers within an autonomous system [2]. To run the protocol, each router counts the number of bytes that pass through its neighboring routers. Every so often at the end of a round, the routers report their counter values to each other and each router checks its neighbors for conservation of flow. While this approach is attractive for monitoring within a small autonomous system, it requires that each router keep track of a significant number of counter variables. In the most simple implementation of the protocol, in which misrouting of packets and collusion between multiple bad routers is ignored, WATCHERS requires that each router maintain 6 counters per one-hop neighbor. Furthermore, to check each one-hop neighbor for conservation of flow, a router must obtain counters from all of its two-hop neighbors.

In order to determine whether or not a protocol similar to WATCHERS would be feasible for monitoring router misbehaving between ASes, we first needed to develop an understanding of the topology of the ASes that comprise the Internet. Using BGP routing tables from the Route Views project in Oregon, we wrote a program that determines the number of one-hop and two-hop neighbors for each AS that participates in Route Views [4]. For our analysis, we chose to use data from four different dates spread roughly six months apart, beginning in April of 2001 and ending in September of 2002. As column two of Table 1 demonstrates, each AS router has an average of approximately 4 one-hop neighbors but close to 1,000 two-hop neighbors. This result is surprising; one would expect that the number of two-hop neighbors would be roughly equal to the square of the number of one-hop neighbors. However, our study of AS topology demonstrates that ASes are arranged in a strongly hierarchical manner. Consequently, most ASes are relatively small and connect to only one or two other ASes. However, the ASes to
which these small ASes connect tend to be quite large. Consequently, a local ISP or a medium-sized company running an AS can have a single one-hop neighbor and yet still be able to reach over 1,000 other ASes within two hops.

<table>
<thead>
<tr>
<th></th>
<th>One Hop</th>
<th>Two Hop</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>mean neighbors</td>
<td>median neighbors</td>
</tr>
<tr>
<td>4/19/2001</td>
<td>4.16</td>
<td>2</td>
</tr>
<tr>
<td>9/30/2001</td>
<td>4.15</td>
<td>2</td>
</tr>
<tr>
<td>3/31/2002</td>
<td>4.19</td>
<td>2</td>
</tr>
<tr>
<td>9/29/2002</td>
<td>4.12</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1: Measure of one-hop and two-hop neighbors for all ASes in Route Views

Based on the analysis of AS topology in the Internet, we conclude that it would not be feasible use a protocol similar to WATCHERS to monitor misbehavior between ASes. Because some routers have hundreds of one-hop neighbors and potentially thousands of two-hop neighbors, the WATCHERS protocol requires each router to keep track of too many counter variables. Additionally, WATCHERS would create a serious problem with congestion at the conclusion of every round, because each router would have to exchange counter variables with thousands of two-hop neighbors all at the same time. Perhaps more importantly, however, is the problem of incentive. In an AS, each router is controlled by a single administrative authority. In this situation it is easy to make every router cooperate with the WATCHERS protocol. However, in the Internet, in which nearly every AS is controlled by a different administrative authority, it would be extremely difficult to convince everyone to participate. As a result, we propose a tester node architecture that does not require as much state as previous approaches and does not rely upon the participation of every AS.

3. The ISPY Architecture

ISPY involves two components: a set of secret tester nodes scattered throughout the Internet and a distribution center that responds to queries for the address of a single tester node.
As shown in Figure 1, when AS $X$ wants to test a neighbor $Y$ for misbehavior, $X$ contacts the distribution center and requests the address of a tester node. The distribution center randomly chooses one of $k$ tester nodes and sends $X$ the IP address of the node chosen. $X$ then contacts the tester node it was given and informs the node that it would like to begin a round of testing. Once the tester agrees to proceed, $X$ sends the tester a series of packets through AS $Y$ and the tester keeps track of the number of bytes it receives. When $X$ is finished sending packets, it informs the tester node that testing is complete. The tester then sends $X$ a response indicating the number of bytes it received. If the number of bytes received by the tester divided by the number of bytes that were actually sent falls below some predetermined threshold, $X$ informs $Y$ that it failed to provide adequate connectivity to the secret tester node. Since $Y$ is providing a service to $X$, be it through a peering or a transit relationship, it is $Y$’s responsibility to ensure that the problem is fixed. Thus, $Y$ must repeat the procedure with the tester node provided by $X$ to determine if the problem is local to $Y$ or if it lies somewhere further along the path to the tester.
With this architecture in place, there is no longer any incentive for ASes to sell transit service or enter into a peering relationship for which they lack the necessary bandwidth or hardware. If all the ASes are monitoring their neighbors and each AS knows that it is being monitored, financial penalties can easily deter an AS from misbehaving out of greed. Likewise, by counting the number of packets dropped, the system is capable of detecting an AS that is being controlled by an attacker. However, in order to make ISPY immune to intelligent misbehavior, a number of security issues must first be addressed.

We assume that intelligently misbehaving routers are able to read all unencrypted packets and react accordingly to evade detection. For example, if a router can figure out the IP address of a tester, it can forward only those packets destined for the tester and drop all the rest. Therefore, when a router $X$ is testing a neighbor $Y$ with secret tester node $Z$, we have to make sure that $Y$ cannot figure out the identity of $Z$. Likewise, we have to ensure that $Y$ cannot simply contact the distribution center and request the IP addresses of all $k$ secret tester nodes participating in the monitoring system. In order to achieve this goal, each AS must have a public and a private key. Similarly, the distribution center and all the secret tester nodes need both a public and a private key.

When a router $X$ requests the address of a tester, it should sign the message with its private key. This allows the distribution center to verify $X$’s identity and log $X$’s request in order to prevent $X$ from making an unlimited number of requests to discover all $k$ secret nodes. Likewise, when the distribution center responds to $X$, it must sign its response with its private key and encrypt the message with $X$’s public key. Signing the message allows $X$ to ensure the response is genuinely coming from the distribution center, and encrypting it with $X$’s public key prevents $Y$ from intercepting the message and discovering the identity of the secret tester.
To initiate a testing session with the secret node, $X$ must sign the first message to the secret tester with the secret tester’s own public key. This will prevent $Y$ from snooping on the packet and discovering that $X$ is about to initiate testing. Similarly, the reply the in which the secret node agrees to begin testing with $X$ should be encrypted with the secret node’s private key to prevent $Y$ from spying. Thereafter, all packets sent from $X$ to the secret node do not have to be encrypted. However, the last reply from the secret node to $X$ should be signed with the secret node’s private key. This ensures that the response is authentic. Finally, the size, type and number of packets that $X$ sends to the secret tester should be randomized in order to prevent $Y$ from detecting any patterns in the communication session. This will also take away the incentive for a rogue secret node to report that it received more packets than it really did.

4. Number and Placement of ISPY Tester Nodes

In this section we examine two questions critical to the successful implementation of ISPY: how do we distribute the secret tester nodes and how many tester nodes do we need?

4.1 How do we distribute the secret tester nodes?

In considering the first question, where to place the secret tester nodes, it is important that the nodes not be all bunched together. If the secret nodes are not geographically spread out and they share a common prefix in their IP address, then it is easy enough for a misbehaving router to simply filter out packets with a matching prefix and drop all other packets. However, it is also desirable that the secret nodes not be scattered all over the Internet, where the nodes may be many hops away from a router communicating with them for the purposes of testing. As the distance from a router to a secret node increases, it becomes more difficult to pinpoint the cause of packet loss. Therefore, we want the secret nodes to be somewhat “centrally located” with respect to the topology of the Internet. More specifically, for all ASes in the Internet, we want to
minimize the expected number of hops to some randomly chosen secret tester node. Fortunately, this problem is easily solvable in polynomial time, because we can reduce it to a minimization for some randomly chosen tester node of the distance to all ASes in the Internet. This can be computed with Dijkstra’s algorithm for single-source shortest path trees. For each AS in the Internet, we compute the single-source shortest path tree, assuming all edges connecting ASes are equally weighted, and we sort the ASes in ascending order by tree size. Then, to get the \( k \) best locations, we simply chose the ASes with the \( k \) smallest tree sizes.

4.2 How many tester nodes do we need?

Unfortunately, the answer to the second question, how many secret nodes do we need, is not quite as clear. Of course, with over 20,000 ASes potentially participating in the monitoring system, it is important to choose a \( k \) value large enough so that the secret nodes are not overloaded with requests. Additionally, we want to pick \( k \) large enough so that it is not possible for any one AS or any small group of collaborating ASes to determine all of the secret nodes in the monitoring system. However, there is a tradeoff with making \( k \) very large. As the value of \( k \) increases, so too does the expected number of hops from some randomly chosen secret node to all routers in the Internet. This tradeoff is summarized in Figure 2. With only 50 secret tester nodes, the expected number of hops between a randomly chosen secret node and a router is approximately 2.55. However, as \( k \) is increased to 500, this distance increases to roughly 2.90.
5. Conclusion and Future Work

The Internet is a crucial component of our nation’s communication infrastructure. In order to make it less prone to attack and abuse, we need mechanisms in place that are capable of detecting misbehavior. In this paper, we present the ISPY system which is capable of identifying malicious routers in a wired network, and we examine the security measures needed to prevent intelligently misbehaving routers from evading detection. Additionally, we analyze ISPY’s architecture to determine the optimal placement of the secret tester nodes and we evaluate the tradeoff between the number of secret tester nodes in use and the distance from a router to a tester. We believe that ISPY represents an improvement over previous approaches, because it requires less state and it does not require the explicit participation of every AS in the Internet.

In future work, we plan on examining ways to make the secret node distribution center less prone to attack. Currently, it represents a single point of failure in our architecture. However, the simple solution of adding redundant distribution centers won’t work, because it compromises the security of the ISPY system. If there are too many distribution centers, a group of malicious routers could query all the centers simultaneously and discover the location of a
large subset of the secret tester nodes. Additionally, we intend on conducting a more rigorous analysis of how many tester nodes are needed.

References


