Reconstruction of Malicious Internet Flows

Omer Demir
Graduate Center, City
University of New York
(CUNY) New York, NY, U.S.A.
odemir@gc.cuny.edu

Bilal Khan
John Jay College, City
University of New York
(CUNY) New York, NY, U.S.A.
bkhan@jjay.cuny.edu

Ala Al-Fuqaha
Computer Science Dept.
Western Michigan University
Kalamazoo, MI, U.S.A.
ala@ieee.org

ABSTRACT

We describe a general-purpose distributed system capable of traceback of malicious flow trajectories in the wide area despite possible source IP spoofing. Our system requires the placement of agents on a subset of the inter-autonomous system (AS) links of the Internet. Agents are instrumented with a uniform notion of *attack criterion*. Deployed, these agents implement a self-organizing, decentralized mechanism that is capable of reconstructing topological and temporal information about malicious flows. For example, when the attack criterion is taken to be based on excessive TCP connection establishment traffic to a destination, the system becomes a traceback service for distributed denial of service (DDoS) attacks. As another special case, when the attack criterion is taken to be based on malicious payload signature match as defined by an intrusion detection system (IDS), the agents provide a service for tracing malware propagation pathways. The main contribution of this paper, is to demonstrate that the proposed system is effective at recovering malicious flow structure even at moderate levels of deployment in large networks, including within the present Internet topology.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General— *Security and protection*

General Terms

Security.

Keywords

Distributed denial of service, flow reconstruction.

1. INTRODUCTION

We define *flow reconstruction* as "determining the true sources and/or routes of packets going to a given destination". There are many obstacles to flow reconstruction arising from the design of Internet architecture itself; here we give only a brief summary – a more detailed treatment is given in [6]. First, in order to gain the most of the Internet, its network link resources are shared among all users. Unfortunately, there is no intrinsic enforcement that the sharing is fair. Second, the network core must deal with very high volumes of traffic which must be processed fast, so core network components do as little as possible per packet, requiring all complex computations to be at the edge computers and thereby leaving a core that is unable to provide enhanced transport services. Finally, the Internet is composed of interconnected networks managed by heterogeneous authorities making widespread deployment of defense mechanisms difficult.

Given the aforementioned inherent obstacles, the notion of "solving" the flow reconstruction problem can have many interpretations. Here we focus on the problem of determining the true origins and mechanics of malicious flows to a given destination, where maliciousness is determined by a well-defined attack criterion. We will consider two special cases of *attack criterion* for concreteness:

- **DDoS**: An agent declares the attack criterion met when it observes excessive TCP connection establishment traffic to a destination. DDoS attacks have been identified as the most critical concern by the Internet Service Providers in Arbor Network's 2008 survey [1].

- **Malware**: An agent declares the attack criterion met when packet payload matches a malware signature.

In general any attack citerion may be used, provided it is applied uniformly to all agents and it satisfies the following condition:

> If an agent A upon observing packets P going to victim v, decides that the attack criterion has been met, then the attack criterion will also be met at all agents further downstream from A (towards v) when they receive packets P.

Identifying the source of malicious packets is difficult because the source address field in the IP packet header of packets is easy to "spoof". The problem is further rendered intractable by the fact that current network standards do not require network devices to maintain information about the paths which packets take. There have been different approaches to this problem and we give a brief synopsis of prominent examples of each of these approaches below.

2. RELATED WORK

Hash-Based Traceback was introduced by Snoeren and Alex [12], whose system consists of data generation agents (DGA), SPIE collection and reduction agents (SCAR). Each DGA uses k fixed n-bit hash functions of the IP packet headers, to store the packet "digest" in a Bloom filter [4]: Initially (and periodically) a 2^n-bit array is initialized zero. The array is marked at indices corresponding to the each packet seen. When an IP Traceback request arrives, it specifies the time, and copy of the packet to be traced. The SCARs then ask the routers to determine whether the packet was seen using their local Bloom filter databases. The main drawback of hash-based IP traceback is that in order to perform traceback, a sample malicious packet must be presented to the system as soon as the attack starts, since Bloom filter tables might be discarded and the records lost. Also, Hash Based Traceback requires coordination and management.

Packet Marking relies on routers adding identification information to randomly selected packets that they forward, so as to reveal the path the packets have taken [2]. Router information is written into the IP packet's identification field. In order to find the full reverse path, there has to be at least one packet marked by the router closest to the attacker. The main limitation of packet marking is that path convergence is slow. Park and Lee [3] showed that PPM is only effective at localizing the attack origin in single-source attacks. Also, Packet Marking requires overloading the semantics IP packet fields, and entails an increase in router packet processing load.

Active Interaction is a strategy of interfering with malicious traffic to deduce information about attack sources based on the systemic reaction to the interference. Backscatter is the prototypical example of this technique [9], operating at the level of BGP level routers. The Backscatter server announces itself as the destination for spoofed IPs being used as source addresses in the attack. It then sends a BGP route announcement to make the destination network being attacked unreachable. This causes the ingress routers reply with a "Destination unreachable" message to the source IP of the malicious packets, thus revealing their entry point to the backbone. While innovative, Backscatter has many drawbacks, most notably, a huge collateral effect in which legitimate traffic to the victim is blocked at the BGP level, and requiring modifications to the BGP protocol.

3. SYSTEM DESIGN

We sought to develop an agent-based system capable of collecting structural and temporal data concerning malicious flows, even at modest levels of agent deployment. We required that the system (i) require no centralized coordination, unlike e.g. [12]; (ii) not entail router modification or significantly increase router processing load (in contrast to e.g., [2, 11]); (iii) be built using existing Internet protocols (as distinct from e.g., [5, 9]).

We begin by giving an informal description of the agent architecture and its operation. Each agent may be viewed as a device which resides on inter-AS links. Each agent can (i) passively listen to ingress/egress transport layer traffic on a router port, and (ii) optionally injects ICMP and TCP layer control traffic into the stream. The agent operates by passively listening to traffic, by sampling packet headers and aggregating statistics based on destination IP address. In
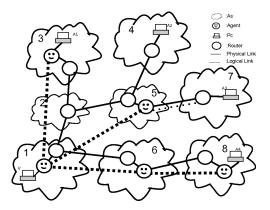

Figure 1: Sample Flow Reconstruction

addition, the agent listens to all ICMP reply packets (header and payload) regardless of their destination. Whenever a router determines that traffic to a destination IP (known hereafter as the "victim") satisfies the attack criterion, it triggers an alarm function, which causes the agent to create an Alert to Downstream (AD) message and send it towards the victim. The purpose of this message is two-fold: (i) it informs downstream agents about a hypothesized attack on the victim, and (ii) it initiates the formation of a logical link in an agent overlay network specifically instantiated in response to the attack. The AD message is implemented as the payload of ICMP reply packet, whose destination address is the victim's IP address; it is sent by starting with a TTL of 1, and incrementing the TTL gradually until an acknowledgment is received from the next downstream agent in the direction of the victim. Whenever an agent sees an AD message in an ICMP reply packet, it replies with an acknowledgment, revealing itself to be the next downstream agent in the direction of the victim, and causing the upstream agent to terminate its TTL-increasing search process. The two agents can then share information concerning the attack on the victim over this newly formed logical link. If we view each logical link as a directed edge from an upstream agent to its downstream agent, the resulting logical network yields a distributed representation of a tree that is a maximal solution of the flow reconstruction problem corresponding to the scenario at hand. Information about the structure of this overlay network can be queried in real time by sending a broadcast message in the overlay network. Agents store their incident logical adjacencies in a distributed database that can be recursively queried to output the structure of malicious flows.

Example. We illustrate malicious flows where v is the victim, clouds numbered 1, 2, 3, 4, 5, 6, 7, 8 represent ASs, A1, A2, A3, A4 represent malicious flow sources, circles represent routers, happy faces represent agents, and straight-lines represent inter-AS link between autonomous systems. Figure 1 shows the reconstructed malicious flow. The path to the attacker in AS-8 is fully reconstructed in full detail. Although the malicious traffic from AS-3 comes through non participating AS-2, the path to the attacker is successfully reconstructed. The attacker from AS-7 can be traced only up to AS-5. The system enables traceback to the extent possible with agent deployment. The reconstructed flows provide the victim with actionable information about both malicious flow structure and dynamics.

4. MATHEMATICAL FORMULATION

Before we can hope to quantify the performance of the proposed system, it is necessary to formally describe the problem that the agents are attempting to solve. Only then can we define the structure of a solution, and performance measures that quantify the solution quality with respect to the problem instance.

4.1 Flow Reconstruction Problem

An instance of the **flow reconstruction problem** (FRP) is a tuple (G, R, A, D, v) where $G = (V, E)$ is a network on nodes V and $E \subset V \times V$ is a set of undirected edges between nodes. The function $R : V \times V \to V$ represents a routing table, where $R(u, d) = v$ is the next hop v on the path from u to d. The set $D \subseteq V$ is a set of attacking nodes which simultaneously attack the victim $v \in V$. The agents have been deployed on a set of links $A \subseteq E$.

To begin, we define **flowstep** $f(d, v, n)$ for every non-negative integer n, inductively, as follows: $f(d, v, 0) = d$, and $f(d, v, n+1) = R(f(d, v, n), v)$. The sequence of flowsteps $F(d, v) = (f(d, v, i); i = 0, 1, ...)$ is called the **flow** from d towards v.

4.2 Formal Solution

A **valid solution** to an FRP is a logical overlay network $L = (S, E_S)$ on a subset of agents $S \subset A$ where $E_S \subset S \times S$ and $\forall e \in S \Rightarrow \exists d \in D \wedge \exists i \in \mathbb{N}(f(d, v, i) = e)$. Additionally, $\forall (e, x) \in E_S \Rightarrow \exists d \in D \wedge \exists i, k \in \mathbb{N}$ where $f(d, v, i) = e$, and $f(d, v, k) = x$, and $\forall j \in (i, k)$ $f(d, v, j) \notin S$.

Stating the above conditions informally: (i) Every agent in the solution set S lies on the flow from some attacker to the victim; (ii) If two agents appear successively in the flow from some attacker to the victim, then these two agents are connected by a logical link in L_S. A solution $L = (S, E_S)$ is said to be **maximum valid** if every agent through which an attacker-originated flow transits, appears in S.

The authors showed previously [7] that for any instance (G, R, A, D, v) of the flow reconstruction problem, there is a unique maximal valid solution. Accordingly, we define the **solution function** s which assigns to each instance (G, R, A, D, v) of the flow reconstruction problem this unique maximal valid solution. Hereafter, we denote the unique maximum valid solution of (G, R, A, D, v) as $s(G, R, A, D, v)$.

4.3 Performance Measures

A performance measure evaluates the quality of a solution (S, E_S) to a problem instance (G, R, A, D, v). We begin by defining a function $d_{G,R,v} : V \times V \to \mathbb{N} \cup \{\infty\}$. Intuitively, $d_{G,R,v}(x, y)$ equals the number of hops that a packet takes to reach y when it is sent by x to v in graph G according to routing table R. Note that $d_{G,R,v}$ is not generally symmetric or transitive, and hence does not define a metric on V. Now given any $Y \subset V$ and $x \in V$, define:

$$d_{G,R,v}(x, Y) = \min_{y \in Y}\{d_{G,R,v}(x, y)\}$$

The set of **undiscovered** attackers $U \subset D$ is defined as the set of vertices u for which $d_{G,R,v}(u, S) = \infty$ We define the performance measure of **undiscovered attacker rate**

$$M1((G, R, A, D, v), (S, E_S)) = \frac{|U|}{|D|}.$$

When M1 is 0 (resp. 1), every (resp. no) malicious flow is

intercepted and hence detected by an agent. Clearly, lower values of M1 are preferred.

The **mean normalized distance to discovered attackers**

$$M3(G, R, A, D, v), (S, E_S) = \frac{1}{|D \setminus U|} \sum_{d \in D \setminus U} \frac{d_{G,R,v}(d, S)}{d_{G,R,v}(d, v)}$$

When M3 is close to zero, every attacking flow that has been intercepted by an agent has been intercepted close to the attacker. In this case, traceback succeeds in getting close to the attack sources. When M3 is close to one, every attacking flow that is intercepted by an agent has been intercepted close to the *victim*. In this case, traceback fails to reach the attack source. Clearly, lower values of M3 are preferred.

Example. In Figure 1, nodes $A1$, $A2$, $A3$, and $A4$ are attacking to victim V. In this case $A1$ is discovered by the agent in cloud 3, $A2$ is discovered by the agent in cloud 1, $A3$ is discovered by the agent in cloud 5, and $A4$ is discovered by the agent in cloud 8. This makes $|U| = 0$. Since $|D| = 4$, in this example $M1 = \frac{0}{4} = 0$, which means all the attackers are discovered. Since $d_{G,R,v}(A1, S) = 0$, $d_{G,R,v}(A1, v) = 3$, $d_{G,R,v}(A2, S) = 4$, $d_{G,R,v}(A2, v) = 4$, $d_{G,R,v}(A3, S) = 1$, $d_{G,R,v}(A3, v) = 4$, $d_{G,R,v}(A4, S) = 0$ and $d_{G,R,v}(A4, v) = 3$, it follows that $M3$ for this example is $M3 = \frac{1}{3}(\frac{0}{3} + \frac{4}{4} + \frac{1}{4} + \frac{0}{3}) = \frac{5}{12}$.

4.4 Expected Performance Measures

Unfortunately, in practice, we do not know where the attackers $D \subset V$ lie in G, nor do we know which victim they will choose to target. In the case of DDoS attacks, for example, botnets are frequently involved, and thus, arbitrary sets of attacking nodes located all over the Internet which collude to attack the chosen victim. Because we do not know the locations of the attackers or victims, the M1 performance measure defined in the previous section cannot be directly computed.

Starting from our definition of M1, we seek a derived performance measure that depends only on G, R, and A. This measure, denoted $E[M1]$, is the expected fraction of attackers which will be discovered when a random set of nodes collude to attack a random victim. Note that the **expected fraction of undiscovered attackers** $E[M1]$ on the triple (G, R, A), can be computed as:
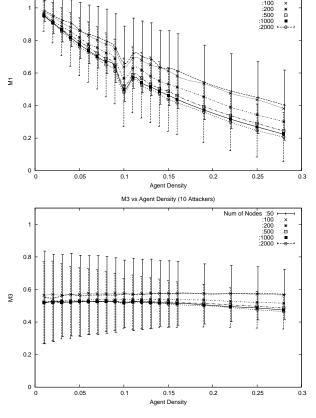
$$\sum_{D \subseteq V, |D|=1, v \in V} \frac{M1((G, R, A, D, v),\ s(G, R, A, D, v))}{|V|^2}$$

Similarly, instead of M3, we use the **expected normalized distance to discovered attackers** denoted $E[M3]$, may be computed for a triple (G, R, A) as follows:

$$\sum_{D \subseteq V, |D|=1, v \in V} \frac{M3((G, R, A, D, v), s(G, R, A, D, v))}{|V|^2 \cdot (1 - E[M1])}$$

5. RESULTS

The simulation creates a randomly generated network with routing tables based on Dijkstra shortest paths; then a set of attackers, a set of agents, and a victim is selected. The aforementioned protocol finds the solution which consists of all agents on paths from each attacker to the victim. Finally, the solution is assessed in terms of performance measures.

M1 vs Agent Density (10 Attackers)

Num of Nodes :50
:100 ×
:200 ✳
:500 □
:1000 ■
:2000 ⊙

M1

Agent Density

M3 vs Agent Density (10 Attackers)

Num of Nodes :50
:100 ×
:200 ✳
:500 □
:1000 ■
:2000 ⊙

M3

Agent Density

Figure 2: M1 (top) & M3 (below) vs. agent density.

- Create a network, $G = (V, E)$, $|V| = n, |E| = m$ and compute R the routing table for G, based on min-hop routing.

- Set the agent density ϵ and attacker density δ.

- For each of 100 agent sets of size ϵm, which are uniformly randomly selected.

- For each of 100 attacker sets of size δn, which are uniformly randomly selected.

- Consider each node v in V, in turn, as the victim.

- Create an instance AFRP problem (G, R, A, D, v).

- Obtain a maximal valid solution and compute M1 and M3 for this solution.

We start creating our network by first creating the nodes by randomly selecting their (x,y) coordinates uniformly at random in a $\sqrt{n} \times \sqrt{n}$ square. Then we make the graph connected by sequentially adding random links, selected among all the possible pair of nodes according to Waxman [14] probabilities. If the graph is not connected when the edge density reaches 2.1, we continue to add edges until the graph becomes connected. Agents are located on the links so we set the agent density with respect to the number of links in the network.

We begin by evaluating our system for connected Waxman networks of sizes 50, 100, 200, 500, 1000 and 2000, constructed according to this procedure. Figure 2 shows that

as the agent density increases from 1% to 15%, the percentage of attackers that remain undiscovered (M1) drops from 90% to under 50%. As the agent deployment grows further to 28%, only 20% of the attackers (on average) are able to evade detection. More important is the fact that for any given deployment, the payoff obtained is greater as the networks under consideration become larger. For example, our experiments showed that with a 15% deployment of agents in a network of 50 nodes, almost 62% of the attackers remained undiscovered–but when the network under consideration grew to 2000 nodes, the same 15% deployment yielded much better performance–only 42% of the attackers evade detection. This analysis indicates that the proposed solution scales, performing increasingly better in larger networks. The error bars in the graph, while large, are also seen to shrink as the network size increases. For example, at 25% deployment, the M1 measure of a 50 node network has a standard deviation of 20%, while for 2000 node networks, the variance shrinks to 14%. The high variances observed are to be expected, since the performance measures are sensitive to the choice of attackers, agents and victim– all of which are random. The lower boundary of the error bars indicate that some agent set placements perform much better than others; describing the criteria under when this occurs is the subject of ongoing research by the authors.

Figure 2 (bottom) depicts the behavior of M3 with respect to agent density. Recall that M3 is the normalized distance between the furthest agent in the reconstructed flow and the nearest corresponding attacker: An M3 value of 0 signifies that traceback was able to reconstruct the flow to the precise autonomous system in which the attacker resides. A value of 0.5 means flow reconstruction was only possible (on average) halfway from the victim to the attacker. The graph shows that the normalized distance to the attacker does not change significantly as deployment increases. More precisely, while M3 decreases slightly as the network size increases from 50 to 500, the extent of the decrease stabilizes for networks much larger than 500. The error bars in the figure show that the variance in M3 decreases as agent density increases, since system performance is less susceptible to the "bad" placement of a few agents. On the whole, the graph shows us that in large networks, even a modest deployment of our system will allow agents to reconstruct the attack flow halfway up to the attacking node.

5.1 Experiments on the Internet Topology

The purpose of the next set of experiments was to determine how our agent based architecture performs on Internet-scale networks. This allowed us to determine scalability and also address possible limitations in the fidelity of the Waxman network model. For this purpose we get autonomous system information from the CAIDA [13] project. We imported the inter-AS topology into our simulation by creating the nodes for autonomous systems and connecting them with links as prescribed. We then repeated the previously described experiments on the Internet using random agent placement.

Figure 3 (top) shows the M1 curve for a random agent placement on two different networks. The first is a Waxman network of 200 nodes and the other one is the CAIDA Internet topology with 17699 nodes. The graph shows us that the curves behave similarly as agent density increases, though the CAIDA network decreases more rapidly than the
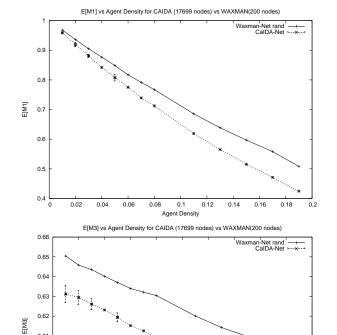
Figure 3: M1 (top) and M3 (below), for agents placed randomly in the Internet.

random network. Figure 3 (bottom) show M3 curve for the same two networks. Again, both curves indicate similar behavior, but the CAIDA network is lower than the curve of the Waxman network, indicating that our scheme scales to large networks and beyond the Waxman model. At a 20% deployment within the Internet, half the attacker flows are represented within the agent tree an we are able to traceback on average halfway from the victim to the attacker.

6. CONCLUSION AND FUTURE WORK

We presented a scalable agent-based system for collecting information about the structure and dynamics of malicious attack flows. The agents in our system are placed on inter-autonomous system (AS) links and implement a self-organizing decentralized mechanism capable of reconstructing topological information about the spatial and temporal structure of attacks. We showed that our system is effective at recovering malicious flow structure, even at moderate levels of agent deployment. Finally, and most significantly, we report that the effectiveness of the system *improves* as network size increases and extends beyond artificial Waxman networks to real-world data Internet topologies. Taken together, these results point to the viability of the proposed system as a scalable solution to the flow reconstruction problem.

Future work. The next stages of our research will involve the development of more sophisticated algorithms for placing agents. While the variance in M1 and M3 measures obtained are quite small, we are hopeful that by carefully placing the agents within the network, we will be able to deploy a system that performs well at even sparser agent densities.

7. REFERENCES

[1] Arbor Networks, `http://www.arbornetworks.com`.

[2] Bellovin, ICMP traceback messages,RFC draft, September '`http: //tools.ietf.org/draft/draft-bellovin-itrace/ draft-bellovin-itrace-00.txt` (2000).

[3] Bellovin, Cert advisory ca-1996-26, Cert Advisory, '`http://www.cert.org/advisories/CA-1996-26.html` (1996).

[4] Bloom, B. H.: Space time trade-offs in hash coding with allowable errors, Commun. ACM, vol. 13, no. 7, pp. 422–426, (1970).

[5] Burch and Hal: Tracing anonymous packets to their approximate source, Proceedings of the 14th USENIX conference on System administration. Berkeley, CA, USA: USENIX Association, 319–328 (2000).

[6] Demir O. : A Survey of Network Denial of Service Attacks and Countermeasures. City University of New York, Computer Science Department. (2009).

[7] Demir, O., Khan, B. : An Agent-based Architecture for Flow Reconstruction of DDoS Attacks. Submitted to International Communications Conference (ICC) 2010, Cape Town, South Africa, 23-27 May 2010.

[8] Demir, O., Khan, B.: Quantifying Distributed System Stability through Simulation A Case Study of an Agent-based System for Flow Reconstruction of DDoS Attacks. In: Proceedings of the 1st Intelligent Systems, Modelling and Simulation Conference, Liverpool, England, 27-29 January 2010.

[9] Gemberling B., Morrow, C., and Greene, B.:ISP security-real world techniques. presentation, nanog. NANOG, `www.nanog.org` (2001).

[10] Gligor V.D.: A Note on Denial-of-Service in Operating Systems. IEEE Trans. Softw. Eng. 10, 320–324 (1984).

[11] Savage, S, Wetherall, D. Karlin, A. and Anderson, T.: Practical network support for IP traceback, SIGCOMM Comput. Commun. Rev., vol. 30, no. 4, pp. 295–306, (2000).

[12] Snoeren, A. C.: Hash-based IP traceback, in SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications. New York, NY, USA: ACM, pp. 3–14, (2001).

[13] The IPv4 Routed /24 AS Links Dataset, Young Hyun, Bradley Huffaker, Dan Andersen, Emile Aben, Matthew Luckie, K. C. Claffy, and Colleen Shannon, 11/15/2009, `http://www.caida.org`.

[14] Waxman, B. M.: Routing of Multipoint Connections.: Broadband Switching: Architectures, Protocols, Design, and Analysis. IEEE Computer Society Press, Los Alamitos, CA, USA (1991).