De-anonymizing the Internet Using Unreliable IDs

Yinglian Xie, Fang Yu, and Martín Abadi
Microsoft Research Silicon Valley

ABSTRACT

Today's Internet is open and anonymous. While it permits free traffic from any host, attackers that generate malicious traffic cannot typically be held accountable. In this paper, we present a system called HostTracker that tracks dynamic bindings between hosts and IP addresses by leveraging application-level data with unreliable IDs. Using a month-long user login trace from a large email provider, we show that HostTracker can attribute most of the activities reliably to the responsible hosts, despite the existence of dynamic IP addresses, proxies, and NATs. With this information, we are able to analyze the host population, to conduct forensic analysis, and also to blacklist malicious hosts dynamically.

Categories and Subject Descriptors

C.2.0 [**Computer Communication Networks**]: General—*security and protection*; C.2.3 [**Computer Communication Networks**]: Network Operations—*network management*

General Terms

Measurement, Security

1. INTRODUCTION

The Internet is designed to be open and anonymous. Just by obtaining an IP address, a host can easily connect to the Internet and freely talk to other hosts without exposing its real identity. This open and anonymous architecture, which enabled the Internet to expand quickly, is also the source of security concerns. Attackers can easily hide their real identities behind IP addresses. Dynamic IP address assignment [10] poses challenges to the commonly used IP-based approach to detect, blacklist, and block malicious traffic. When an attacker changes its IP address, legitimate activities that subsequently use the old IP address will be misclassified as bad, while malicious activities from the new IP address will slip through. The numerous proxies and NAT devices also imply that blacklisting can result in denial of service to many legitimate clients that share IP addresses with attackers.

SIGCOMM'09, August 17–21, 2009, Barcelona, Spain.

Security rests on host accountability. By accountability we mean the ability to identify the hosts responsible for traffic [18], which is typically the basis for punishing misbehavior. Several clean-slate solutions have been proposed to provide accountability in the Internet, thus eliminating the problems created by dynamic IP addresses, proxies, and NATs. For example, Accountable Internet Protocol (AIP) [1] changes the IP layer by utilizing self-certifying addresses to ensure that hosts can prove their identities. Although these proposals are attractive, they are hard to deploy and many service providers need immediate means to combat network attacks.

In this paper, we aim to develop an immediate, practical approach to associate traffic with hosts. We revisit the utility of IP for accountability and seek to understand its limitations. In particular, we aim to quantify the ability to infer the bindings between a host and an IP address (which we call host-IP bindings). Our study attempts to answer the following two questions:

- To what extent can we use IP addresses to track hosts?
- Can we use the binding information between hosts and IP addresses to strengthen network security?

The answers to these questions have implications in numerous security applications. First, the ability to identify a host over time permits attack investigation to trace back to the start of malicious activities and to reveal previously undiscovered ones. Second, it facilitates the task of building more accurate blacklists to both monitor and block ongoing attacks, and prevent new attacks in the future. Furthermore, the knowledge of compromised hosts detected by one application can potentially be shared to benefit other applications, as botnets may be rented out to different attackers over different periods.

We present *HostTracker*, a system that relies on application-level events to automatically infer host-IP bindings. In today's Internet, establishing accurate host-IP bindings is challenging: malicious activities, dynamic IP addresses, proxies, and NATs make it difficult to differentiate activities from distinct hosts. HostTracker leverages IDs derived from application-layer logs in order to create unique host identifiers and to track the bindings of hosts to IP addresses over time. For our study, we employ a month-long user login trace collected at a major Web email service provider. Our findings include:

- We show that, even without built-in host identities, using IP addresses, anonymized user IDs, and their associated events, we can track a large percentage of host activities with high accuracy. Overall, 76% of the events in the application log can be attributed to hosts, and 92% of hosts can be tracked correctly. This result is consistent across many IP-address

ranges, suggesting that tracking host-IP bindings is widely applicable.

- As an application, relying on a set of previously detected botnet email accounts (5.6 million) [38], we can apply the host-IP binding information to identify the compromised hosts and many additional botnet email accounts (12.6 million) with a low false positive rate (0.4%).
- We can also build a host-aware blacklist using the host-IP binding information. This blacklist can help us follow the trail of malicious hosts in order to block attacks (from 20.8 million malicious accounts) in real time without significantly affecting normal user access (with a 0.1% false positive rate).

With this initial evidence, we believe that tracking host-IP bindings holds promise to improve network security in the current Internet. It opens up new opportunities to re-think many existing attack detection and prevention mechanisms, particularly those that use blacklists to block traffic.

The goal of our work is not to develop bullet-proof accountability schemes, but instead to support accountability well enough to serve the needs of applications. In this sense, our approach is complementary to clean-slate proposals. It might be part of a temporary solution until architectural changes can be made to the networking stack, and it might inform the eventual design of those changes.

Despite the value of accountability, we believe that anonymity also has a legitimate place in the Internet (for instance, for enabling forbidden communication in repressive countries). Our findings suggest that "low-tech" anonymity is more fragile than one might have expected. Nevertheless, anonymity can often be achieved with sophisticated tools (e.g., onion routing [31], anonymous remailers, etc.). Those tools do not negate the value of HostTracker for many applications. Further, we hope that our work will contribute to the ongoing debate on the balance between accountability and anonymity in the next generation of IP networks.

2. RELATED WORK

Host accountability in the Internet has long been a topic of substantial interest. A large body of previous work has focused on providing source accountability to identify the true network origin of traffic. Both stepping-stone techniques and source-address spoofing are commonly leveraged to hide attacker identities. A few early efforts have proposed solutions to detect stepping-stone attacks by packet-timing analysis and content analysis [37, 9, 5, 33]. Ingress and egress filtering [12, 16], which have been partially deployed, can prevent source-address spoofing. Other proposed approaches also require changes to the existing routers or the routing infrastructure. Among them, IP-traceback techniques [29, 19, 28, 3, 6] aim to determine the source(s) of packets by storing additional state at routers or marking packets along their paths. Passport [20] validates host source addresses using cryptographic keys.

Many of today's attacks require establishing TCP connections, in which source-address spoofing is difficult. On the other hand, botnet hosts do not need to spoof IP addresses: the transient nature of the attacks and the dynamics of IP-address assignment make it difficult to pinpoint the exact compromised host entities as their IP addresses change. To offer host accountability as a fundamental security property, Accountable Internet protocol (AIP) [1] is a clean-slate design that uses self-certifying addresses to ensure that hosts and domains can prove their identities without relying upon a global trusted authority.

While clean-slate proposals offer attractive long-term solutions, our work targets today's Internet to mitigate security threats until architectural changes can be deployed globally. With the rapid advance of attack detection techniques (e.g., [25, 26, 14, 13, 36]), we focus on the ability of tracking host-IP bindings so that both proactive and reactive measures can be taken once a compromised host is identified by its IP address at a certain time.

In order to block unwanted traffic from compromised hosts, various blacklists are widely used (e.g., [11, 30, 4]). However, since these blacklists all represent hosts by IP address, it remains unclear how to apply them effectively in the presence of dynamic IP addresses and proxies. (Recent studies have shown that a significant fraction of the Internet IP address space is dynamic [35] and that the number of proxies and NATs is non-trivial [7].) By inferring the host-IP bindings, HostTracker potentially provides new possibilities of applying IP blacklists in a more effective way.

More broadly, botnet detection and defense has attracted a lot of attention. Botnet spamming attacks are commonplace. In many of these, each bot host is used to set up a spam email server [8, 24, 2, 15, 35]. Recently, a new botnet spamming attack signed up tens of millions of free Web email accounts for sending spam emails [32]. Applying a blacklist-based approach to this new attack is even more challenging, as legitimate users and attack accounts may log in from a common host.

Finally, remote device fingerprinting is also an active area of research, where several proposals leverage packet-level traffic characteristics or clock skews of a host to generate OS or device fingerprints (e.g., [22, 21, 17]). Exploiting subtle hardware differences can potentially be quite robust, but the accuracy of the existing techniques currently prevents them from being deployed at a large scale. Our approach offers an alternative solution to hardware fingerprinting—application-level IDs might be regarded as host fingerprints (with noise).

3. PROBLEM FORMULATION

Host accountability is often a key component associated with auditing and forensic analysis. We distinguish host accountability from host authentication. Whereas host accountability may only be "after the fact", host authentication typically establishes a trusted identity before any further traffic or requests from the corresponding entity can be received or processed. Accordingly, in this section, we consider the problem of tracking hosts relying on traces of past application-level events.

3.1 Host-Tracking Graph

Given that an IP address is the only directly observable identifier of host network traffic in the current Internet and that a host's IP address can dynamically change, our goal is to perform a posteriori analysis of traffic (e.g., network activity or application request) logs and generate a *host-tracking graph* to infer the bindings between hosts and IP addresses over time.

Figure 1 depicts an example host-tracking graph for three different hosts `A`, `B`, and `C`. Time advances from left to right, with each rectangle representing the binding between a host and an IP address across a time range. In this example, host `A` was bound to IP address IP_1 during time range $[t_1, t_2]$. Later, host `A` moved to IP_2 and IP_5 during time ranges $[t_3, t_4]$ and $[t_5, t_6]$, respectively. We call a *binding window* a time range during which a host is bound to an IP address. In this example, the binding window for `A` at IP_1 is $w = [t_1, t_2]$.

Formally, we define the host-tracking graph $G : H \times T \rightarrow IP$, where H is the space of all hosts on the Internet, T is the space of time, and IP is the IP-address space. For example, if host $h \in H$ is bound to IP address $IP_1 \in IP$ at time $t \in T$, we represent this binding as $G(h, t) = IP_1$. Similarly, we write $G(h, w) = IP_1$

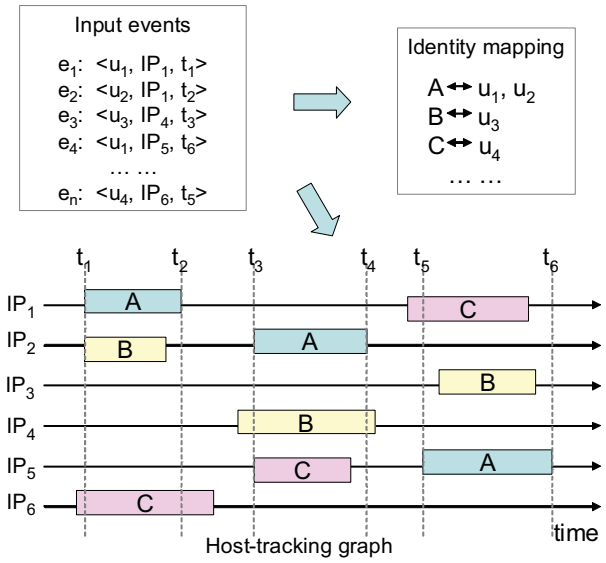


Figure 1: Generation of a host-tracking graph.

when $G(h,t) = IP_1$ for all $t \in w$. With this definition, we do not consider multi-homing hosts where a host may have different IP addresses concurrently. For proxies and NATs, different hosts may concurrently use one IP address. So it is possible to have $G(h_1,t) = IP_1$ and $G(h_2,t) = IP_1$, where $h_1 \neq h_1$. Note that the bindings from hosts to IP addresses may not be continuous, since hosts may not be up all the time and there may also exist unassigned IP addresses. For example, there are no hosts bound to address IP_3 before time t_5 in Figure 1.

With a host-tracking graph, we can attribute input events to the responsible hosts by finding the corresponding host-IP bindings. This enables us to reason about host accountability in two ways. First, for any network traffic originating from an IP address IP_i at time t_j, if there exists a single host h such that $G(h,t_j) = IP_i$, then h may be blamed for such traffic. Second, if host h is known to have bound to a different IP address IP_k at t_j, then we may conclude that h is not responsible for generating any traffic from IP_i at time t_j.

3.2 Host Representation

To derive a host-tracking graph, the first question is how to represent a host. Host representations are often application-dependent. In network security applications, it is desirable to represent a host as *an entire hardware and software stack* and track all its network activity.

Since we lack strong authentication mechanisms, we consider leveraging application-level identifiers such as user email IDs, messenger login IDs, social network IDs, or cookies. These identifiers are fundamentally unreliable in that they do not exactly correspond to hosts. On the other hand, each such identifier is typically associated with a human user. In many cases, such connections are helpful for tracking the corresponding principals. For example, a browser cookie appearing in requests from different IP addresses at different times *may* suggest that the corresponding host has connected to the Internet at different times with different DHCP IP addresses. Therefore, the group of user IDs that is associated with a host can essentially serve as a "virtual ID" for this host in order to keep track of the related activities. Figure 1 shows an example *identity-mapping table* that represents the mappings from unreliable IDs to hosts.

3.3 Goals and Challenges

Our goal is to generate the host-tracking graph using logs with unreliable IDs. As indicated in Figure 1, the input to our problem is a sequence of network or application events $e_1, e_2, \ldots, e_n$ over time. Each event contains three fields: *a unique unreliable ID*, *an IP address*, and the *timestamp* when we observe the ID being associated with the corresponding IP address. We would like to generate two outputs: the first being an identity-mapping table that represents the mappings from unreliable IDs to hosts, and the second being the host-tracking graph that tracks each host's activity across different IP addresses over time.

With the generated host-tracking graph, we refer to the set of input events that can be attributed to certain hosts as *tracked events*, and refer to the remaining events as *untracked events*. Correspondingly, we refer to those IDs that are used to represent hosts in the identify-mapping table as *tracked IDs*, and refer to the group of IDs that all map to one unique host as a *tracked ID group*.

Ideally, with a perfect host-tracking graph, every event is a tracked event. However, the huge amount of noise intrinsic in using unreliable IDs, the existence of dynamic IP addresses, NATs, and proxies, and the presence of malicious attack traffic all make the task of generating the host-tracking graph challenging:

- First, one user and hence one host may be associated with multiple IDs. Furthermore, a group of hosts can also share an ID if they are controlled by a common user.
- Second, a large fraction of the IP addresses in the Internet are dynamic and that their host-IP binding durations have huge variations [35].
- Third, proxies and NATs make it difficult to distinguish individual hosts behind these boxes [7]. Typically we cannot use a fixed set of IDs to represent a large proxy or NAT, as its user population may change frequently. Moreover, proxies and NATs may use dynamic IP addresses as well. Given the limitations of our dataset, we do not further distinguish hosts behind proxies and NATs. However, we do need to identify large proxies and NATs and take special actions in this case. Specifically, we treat proxies and NATs as hosts, and therefore their events are tracked.
- Finally, once a host is compromised, its traffic patterns may differ from those of normal hosts. For example, in recent Web-account attacks, attackers signed up tens of millions of free email IDs and then aggressively used them to send spam from compromised hosts. Those malicious IDs are not useful for identifying compromised hosts when we generate the host-tracking graph.

Therefore, realistic algorithms may generate only a subset of the real-world bindings. Moreover, not all host-IP bindings may be correctly identified. Our goal is to maximize the number of tracked events, while ensuring the host-IP binding inference accuracy.

4. TRACKING HOST ACTIVITIES

In this section, we present *HostTracker*—a system for tracking host-IP bindings by mining application-level activity logs. To simplify our description, in the rest of this section, we use user email IDs as our example application IDs and adopt user-login events as example input events.

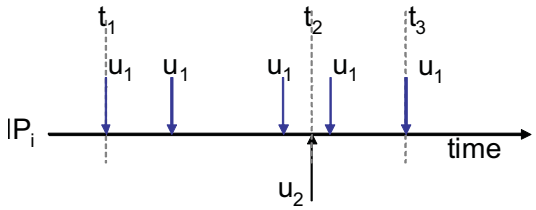


Figure 2: An example set of input events.

4.1 Overview

The host-tracking is performed separately for each IP range. The range information can be derived from the BGP table prefixes [27] or other sources such as the Whois database [34]. Usually, within a range, a user ID may bind to multiple IP addresses from a single host. For user IDs that appear across multiple ranges, the bindings to multiple IP addresses may also be caused by user mobility rather than host mobility. Therefore, HostTracker analyzes events from each range independently. In Section 5.4, we further study user mobility on a global scale.

In the host-tracking process, a crucial piece of information to recover is the identity-mapping table as shown in Figure 1, i.e., which user IDs map to which hosts. Without prior knowledge of which subset of user IDs can potentially be tracked, we regard the identity-mapping table as a hidden model that we would like to estimate. The high-level idea of HostTracker is to iteratively update our estimations so that it best fits the actual observation of the events from the input data.

Let us begin by assuming that each user ID would be a tracked ID group that maps to a unique host. Figure 2 shows an example input of six user-login events (represented as vertical arrows) from users u_1 and u_2 at IP$_i$. Here u_1 logged in five times from time t_1 to t_3. u_2 logged in at the same IP address at time t_2. In our dataset, we observed that the IP-address assignment in a large IP range was either random (e.g., [10, 23]) or static. In the random address assignment case, the probability of a host obtaining the exact same IP address in two different connections in a network range is usually very small. Therefore it is highly likely that the host corresponding to u_1 is bound to IP$_i$ throughout the duration between t_1 and t_3, including t_2. However, under the assumption that each user uniquely maps to a different host, there exists another host corresponding to u_2, which also binds to the same IP address IP$_i$ at time t_2. Our naive one-to-one identity mappings thus lead to an inconsistency, suggesting that we should re-estimate the mapping table.

To do so, we consider three possibilities: (1) multiple user IDs share a host, and in this case u_1 and u_2 both map to one common host (e.g., used by two family members); (2) IP$_i$ is a proxy associated with multiple hosts; (3) u_2 is a guest user (either a real guest account, or introduced by an attacker) to the host associated with u_1, or vice versa. The challenge is how to differentiate the three cases based on available observations. During this process, we would like to maximize the number of tracked events with as few inconsistent events as possible.

More generally, Figure 3 shows the processing flow of HostTracker. At all times, HostTracker maintains a set of ID groups. At the very beginning, HostTracker uses a probabilistic model to initialize the set of tracked ID groups by grouping user IDs that were highly correlated in their login patterns (Section 4.2). HostTracker then applies an iterative refinement process to update the estimated model. HostTracker constructs a host-tracking graph using the current estimations and detects inconsistencies (Section 4.3). Later, HostTracker applies a separate process to remove inconsistencies (Section 4.4). Finally, HostTracker updates the ID groupings, and the updated ID groups are then fed back into the graph construction process again in an iterative way, until the estimated model converges (Section 4.5). The final output is a host-tracking graph with the corresponding identity-mapping table.

4.2 Application-ID Grouping

To derive host-IP bindings using application IDs, our first step is to compute an initial set of ID groups so that each group, with high probability, is associated with one common host. To do so, we compute the probability of multiple user IDs logging in nearby in time from a common host based on their login events.

Intuitively, if a group of users all logged in from one host, their login events will show strong correlations, i.e., appearing at a similar set of IP addresses at nearby times. For example, if user u_1 logged in 10 times at 10 different IP addresses, and u_2 logged in 9 times, and among these login events, user u_1 and u_2 appeared 8 times next to each other at 8 different IP addresses, then they are very likely to share one host.

However, by random IP address assignment, two unrelated users might also login consecutively if they happened to have connected to the Internet consecutively at a common IP address. In particular, in the proxy case, two different users repeatedly using a common proxy might also have a large number of close-by login events.

To quantitatively compute the probability of two independent user IDs u_1 and u_2 appearing consecutively, let us assume that each host's connection (hence the corresponding user login) to the Internet is a random, independent event. Given a sequence of login events from a user u_1, the number of times another user u_2 logged in right before or after u_1 in time follows a binomial distribution. Specifically, let n_1 denote the number of neighboring login slots to u_1, then the probability of u_2 appearing k or more times in these slots can be approximated as follows:

$$P(u_1, u_2) = 1 - \sum_{i=0}^{k-1} \binom{n_1}{i} p_2^i (1-p_2)^{n_1 - i}$$

Here p_2 is the probability of u_2 logging in. With a total of c login events and a total of c_2 login events from u_2, we can approximate p_2 by $\frac{c_2}{c}$. A very small $P(u_1, u_2)$ means there is very little chance for u_1 and u_2 to log in consecutively k times if they are independent, suggesting that u_1 and u_2 might be correlated instead.

HostTracker identifies all pairs of users who have consecutive logins for every IP address. If a pair of user (u_1, u_2) have at least two consecutive logins across all IP addresses in a range (regardless whether the two consecutive logins happened at a same IP address or at two different IP addresses), HostTracker performs this correlation test for this pair, and selects (u_1, u_2) as a correlated user ID pair if $P(u_1, u_2)$ is smaller than a pre-set threshold. For example, if user u_2 logged in 62 times out of 133 of total logins, then $p_2 = 0.47$. In this case, if u_1 and u_2 logged in together 9 times out of a total of $n_1 = 49$ neighboring slots for u_1, then the probability of them appearing together in a random assignment is as high as 0.87. Hence we should not group the two users in this case. Currently, we set the probability threshold to 0.05, but the threshold could perhaps be more systematically tuned based on the login event patterns from each IP range.

Once HostTracker identifies all pairs of correlated users, it further groups them. If user ID pair (u_1, u_2) and (u_2, u_3) both are correlated pairs, then HostTracker groups all three users into a correlated user set $\{u_1, u_2, u_3\}$. This process continues, until there are no more groups that can be further expanded.

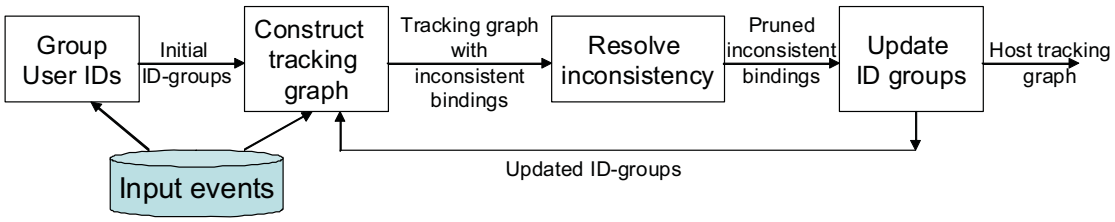


Figure 3: The processing flow of HostTracker.

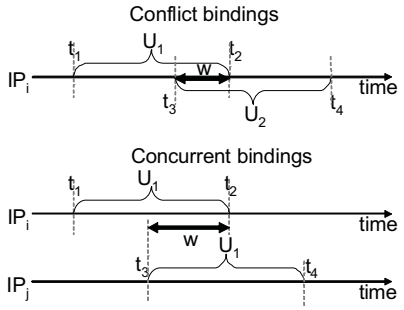


Figure 4: Examples of conflict bindings and concurrent bindings.

At this stage, each expanded ID group is regarded as tracked. For the remaining user IDs that cannot be grouped, HostTracker regards each user as a tracked ID group if the user has logged in at least twice throughout the trace duration. Both sets are merged together for further analysis. In the next few steps, HostTracker iteratively prunes and updates this set.

4.3 Host-Tracking Graph Construction

HostTracker regards each ID group as a candidate host identifier. It constructs a host-tracking graph using these groups. The first operation of this step is relatively simple: for each group U, HostTracker finds the first timestamp t_1 and the last timestamp t_2 at which any user from U logged in at IP address IP_i, and lets $w = [t_1, t_2]$ be the binding window for U at IP_i. We indicate the binding by $G(U, w) = IP_i$.

The second operation is to mark all inconsistent bindings on the graph. There are two types of inconsistent bindings (shown in Figure 4):

- **Conflict bindings:** Two user groups concurrently used the same IP address. For any two bindings $G(U_1, w_1) = IP_i$ and $G(U_2, w_2) = IP_i$, where $w_1 = [t_1, t_2]$ and $w_2 = [t_3, t_4]$. If $t_3 \leq t_2$ and $t_1 \leq t_4$, HostTracker first identifies the overlapped time range w between w_1 and w_2. It then marks both $G(U_1, w) = IP_i$ and $G(U_2, w) = IP_i$ as conflict bindings.
- **Concurrent bindings:** A user group concurrently used two different IP addresses. Specifically, consider two bindings $G(U_1, w_1) = \text{IP}_i$, where $w_1 = [t_1, t_2]$, and $G(U_1, w_2) = \text{IP}_j$, where $w_2 = [t_3, t_4]$. If $\text{IP}_i \neq IP_j$ and w_1 and w_2 overlap, then similarly to the conflict binding case, HostTracker finds the overlapped time range w, and marks both $G(U_1, w) = IP_i$ and $G(U_1, w) = IP_j$ as concurrent bindings.

In the next section, we describe how HostTracker iteratively resolves these inconsistent bindings.

4.4 Resolving Inconsistency

Inconsistent bindings can be caused by the existence of NATs and proxies, guest login events from IDs that cannot be eventually tracked, or incorrect groupings. In this section, we go through these three cases sequentially.

4.4.1 Proxy Identification

Proxy identification can resolve conflict bindings. Since large proxies or NATs allow hosts to concurrently access the Internet through them, they will generate a large number of conflict bindings. Depending on the network configurations, proxies/NATs can use either static or dynamic IP addresses. For example, a DSL-based Internet café may change its IP addresses from time to time.

To find both types of proxies/NATs, HostTracker gradually expands all the overlapped conflict binding windows associated with a common IP address. The purpose of such expansion is to obtain a maximum continuous *conflict window* with a large number of conflict bindings. Specifically, if $G(U_1, w_1) = \text{IP}_i$ and $G(U_2, w_2) = \text{IP}_i$, and the conflict windows w_1 and w_2 overlap, then HostTracker merges the two windows into an expanded conflict window.

For each expanded conflict window, HostTracker checks the degree of event concurrency. It attributes the conflict window to a proxy if the rate of new user arrivals is greater than α and the rate of conflict bindings is higher than β. In our current implementation, we empirically set α to 1 new user per 5 minutes and β to 1 conflict binding per 30 minutes.

After HostTracker identifies a binding window for a proxy, it marks all events falling into the window as proxy events. Recall that we treat proxy events as tracked events, with the corresponding proxy as their origin. With the limitations of our datasets, we do not further distinguish individual hosts using the proxies. In practice, this step can reduce the number of conflict bindings by more than 90%.

4.4.2 Guest Removal

Guest removal is helpful for resolving conflict bindings too. After we mark the user IDs as potentially "tracked" or "untracked" in our initial estimation, it is possible that a tracked group and an untracked ID concurrently appear at the same IP address, resulting in conflict bindings. There are three possibilities in this case: (1) the login of an untracked user ID is a guest login event, and in this case the tracked user group represents the correct host; (2) the untracked ID and the tracked group share a host, but they were not grouped in our first step because of the infrequent logins from the untracked ID; (3) in contrast to case (1), the login event of the tracked user group is a guest event.

In this case, HostTracker uniformly treats events from the untracked group as guest events to resolve the conflict. In both (1) and (2), the guest login events from the untracked ID will be cor-

rectly attributed to the host corresponding to the tracked group. In practice, we found (3) to be rare.

Overall, conflict bindings caused by guest events represent only a small fraction of the total inconsistent events. Although guest events are relatively few (see Section 5.2), it is still important to identify the hosts responsible for them. In our applications, we found most of the guest events to be malicious events on compromised hosts.

4.4.3 *Splitting Groups*

Splitting groups can resolve concurrent bindings. If a user ID group has a large number of concurrent bindings, the initial grouping step might be overly aggressive in grouping IDs from different hosts together. So for each group that had more than one concurrent binding, HostTracker adjusts the grouping by splitting the subset of IDs that caused concurrent login events into a different ID group. For each concurrent binding, HostTracker splits the group and examines whether there still exists concurrent bindings afterwards. If so, it continues the splitting process until there exists no more concurrent bindings for the IDs in the original group. In practice, concurrent bindings are a small portion of all the conflicts, and not many groups are affected in this step.

4.5 Closing the Loop

With the knowledge of proxies, guest events, and the split groups, HostTracker re-estimates the initial identity mappings by removing groups whose member IDs correspond to proxy-only users and guest-only users. Therefore, these IDs become untracked.

This process may appear straightforward. However, the order of the pruned users will affect the final set of remaining tracked IDs. If HostTracker incorrectly identified a trackable user as a guest, then all its events would be regarded as untracked. Given that our goal is to maximize the number of tracked events, only groups appearing at proxies are pruned initially. HostTracker can then iteratively refine the remaining groups following the steps of Section 4.4. In practice, for most IP ranges, the number of tracked groups converges after 4 to 6 iterations.

At this point, it is difficult to further resolve the remaining inconsistent bindings without additional information. To be conservative, HostTracker discards all the events from these inconsistent bindings and treats them as untracked events.

For all the identified host-IP bindings, their actual binding windows may be much longer than the binding windows derived from the login events only. So HostTracker expands their window sizes to increase the coverage. As illustrated by Figure 5, for a host-IP binding $G(A, [t_1, t_2]) = \text{IP}_i$, its window expansion is subject to two constraints: it cannot expand beyond the boundaries of the previous and next binding windows of the same host, nor can it expand beyond the boundaries of the previous and next binding windows on the same IP address. Under the two constraints, HostTracker opportunistically increases the binding-window size by one hour both forward and backward in time. In the case where the time between neighboring events is smaller than 1 hour, HostTracker divides the time gap equally to expand both binding windows.

Finally, HostTracker outputs both the identity-mapping table and the host-tracking graph with expanded binding windows. It classifies Events associated with the identified host-IP bindings are classified as tracked events. For host-IP bindings of a proxy, the corresponding events can be attributed to the proxy that originated the traffic. The remaining untracked events are either non-guest events from untracked IDs, or events within the unresolved inconsistent bindings (those discarded).

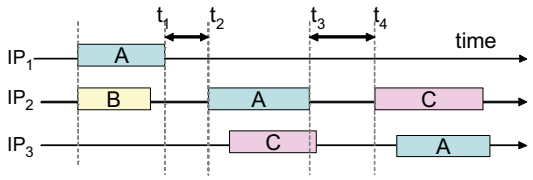


Figure 5: An example of binding-window expansion, where the host-IP binding is $G(A, [t_2, t_3]) = IP_2$. The window can maximally expand from t_1 to t_4.

5. HOST-TRACKING RESULTS AND VALIDATIONS

We applied HostTracker on traces collected at a major Web-email service provider. This section presents the host-tracking results using this dataset. We also evaluate the results using a separate software-update dataset that contains information on the actual host hardware IDs. Our study focuses on understanding the following aspects of tracking host-IP bindings:

- **Coverage:** What percentage of events or time duration can be associated with tracked hosts? How many hosts can be tracked at each network? We find that a large percentage of events can be tracked. The high percentage of tracked events is consistent across most of the IP ranges.

- **Accuracy:** How accurate are the bindings between IP addresses and hosts? Even with dynamic IP addresses and proxies being widely deployed, HostTracker achieves 92%–96% accuracy in our study.

- **Trackable user characteristics:** Can users be tracked across different locations? We find that the majority of the users can be tracked in only one or two network ranges. Highly mobile users that cannot be tracked at all are more suspicious in their email-sending patterns.

5.1 Input Dataset

The input dataset to HostTracker is a month-long user-login trace collected at a large Web-email service provider in October, 2008. The input data volume is about 330 GB, with each entry having three fields: (1) an anonymized user ID, (2) the IP address that was used to perform email login, and (3) the timestamp of the login event. In the log, we observe over 550 million unique user IDs and more than 220 million unique IP addresses. A large percentage of this user population consists of home users that access the Internet from dynamic IP-address ranges.

To obtain IP-address range information, we used the BGP-prefix table collected for the same period. For host-tracking to be statistically meaningful, we discard inactive IP ranges with fewer than 100 events or active for fewer than 7 days during the entire month, and we apply HostTracker to the remaining 30K ranges.

To evaluate the host-tracking results, ideally we would like to obtain the ground truth of the actual host-IP bindings. Without such information available, we adopt a month-long software-update log collected by a global software provider during the same period of October, 2008. The data entries we used for our validation include a unique hardware ID for each remote host that performs an update, the IP address of the remote host, and the software update timestamp.

	Event coverage	IP-day duration coverage
Accumulated	76.0%	79.3%
Median	74.7%	77.7%

Table 1: Percentage of tracked events and IP-day durations for all ranges.

	Regular events	Proxy events	Guest events
Accumulated	85.8%	11.8%	2.4%
Average	80.0%	13.0%	1.2%

Table 2: Breakdown of the three categories of tracked events.

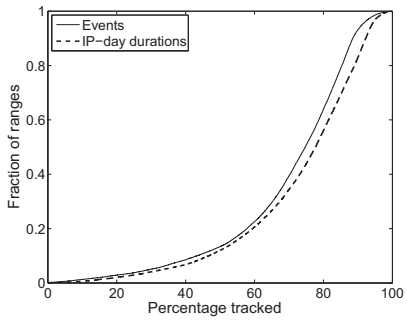


Figure 6: The percentage of tracked events and IP-day durations.

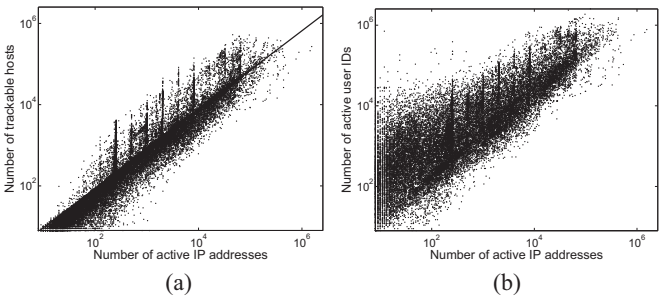


Figure 7: Tracked hosts vs. active IP addresses for each range.

5.2 Tracking Coverage

Using HostTracker on our dataset, we tracked about 220 million hosts. We examine the host-tracking coverage using two metrics: *event coverage* and *IP-day duration coverage*. The *event coverage* is computed as the percentage of events that can be tracked by HostTracker over the total number of input events. Since an IP address may not always be actively used (i.e., assigned to a host), we aim to quantify the inferred host-IP binding durations. We divide time into days. If all the events of an IP address on a day are tracked, then we consider the entire IP-day pair to be tracked. The *IP-day duration coverage* quantifies this coverage and is computed as the the percentage of tracked IP-day pairs over the total number of IP-day pairs in the input.

Table 1 shows both the accumulated coverage and the median coverage across all ranges. In aggregate, HostTracker identifies a total of 76% of the input login events as tracked, with a median coverage of 74.7% for individual ranges. The IP-day duration coverage is slightly higher, about 79% in the aggregated case, and 77.7% in the median case. The percentage of tracked events and IP durations is remarkably high.

Furthermore, we find that a large percentage of IP ranges have a consistent high fraction of tracked events and IP durations. Figure 6 shows the cumulative distribution of both coverage metrics across different IP ranges. Both curves have similar distributions: about 88% of ranges have at least 50% of tracked events and IP-day durations, suggesting that a large percentage of such application traffic can be attributed. However, highly trackable ranges are a relatively small portion: about 7% of ranges have more than 90% of tracked events, and about 18% of ranges have more than 90% of tracked IP-day durations.

We also examine the breakdown of tracked events across three different categories in Table 2. In total, about 86% of all tracked events are classified as regular user login events. Proxy events are also a non-trivial fraction, accounting for roughly 12% of all tracked events. As expected, guest login events are infrequent and are a small fraction (2.4%).

We proceed to study the number of tracked hosts in each IP address range. Figure 7 (a) shows the scatter plot of the tracked host-population size vs. the number of active IP addresses observed from each address range. Interestingly, we find that the number of tracked hosts roughly follows a linear distribution to the number of actively used IPs. Using linear regression, the linear slope is about 0.64. (The figure is plotted in a log-log scale, but the linear relationship holds for the non-scaled data.) Such linear relationship suggests it might be possible to infer the actual host population based on IP address activity.

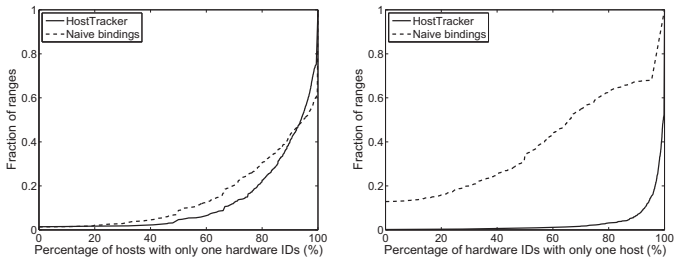
We also notice a few spikes around popular network-range sizes 256, 1024, 4096, etc. Manual investigation shows that a majority of these ranges are geographically located together in one country in South America. Thus they may be configured similarly for high network utility. We also plot the user-ID population size vs. the number of active IPs in Figure 7 (b); the linear correlations are not as strong.

5.3 Validation

Since software updates are relatively infrequent events, and not all computers perform updates, using the software-update data itself to track host-IP bindings will generate very small and possibly biased coverage in the Internet. In our validation, we consider only the set of hosts (represented by ID groups) that each had at least two software update events in their host-IP binding windows. We can evaluate about 357K of the tracked hosts using the software-update dataset. Although this represents only a small fraction of the tracked hosts, and the hosts are not picked randomly, the validation results are encouraging.

For each host to be evaluated, if HostTracker correctly tracks its host-IP bindings, we expect there will be only one unique hardware ID pertaining to the updates. On the other hand, if the corresponding hardware ID had any update events, we expect these events to be also associated with the exact same host as well. Overall 92% of the evaluated hosts each corresponded to only one hardware ID. Similarly, 96% of the hardware IDs observed in our evaluation corresponded to only one host. For our evaluated IP ranges, about 74% of the ranges were likely dynamic-IP ranges where hosts switch their IP addresses. Despite their use of dynamic IP addresses, our evaluation results suggest that a large fraction of the hosts in these ranges can be tracked correctly.

Figure 8 plots the CDF of the tracking accuracies across all ranges. We compare HostTracker with a naive method where we straightforwardly bind IP addresses based on user IDs. This naive method



(a) (b)

Figure 8: Cumulative distributions of the validation results using the software-update data.

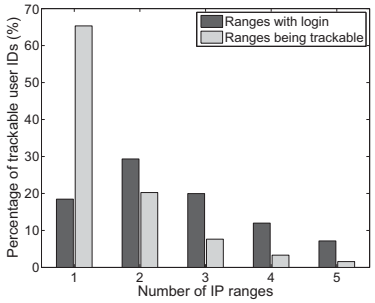


Figure 9: Histogram comparison of the tracked user population: the number of IP ranges they logged in vs. the number of IP ranges in which they are classified as being tracked.

is conservative in the majority of cases by treating events associated with only one user ID as a host ID. Although HostTracker and the naive method achieved comparable accuracies in associating user IDs to hardware IDs, HostTracker significantly outperforms the naive method in terms of the accuracy of associating hardware IDs with host IDs. For the naive method, over 40% of the ranges have less than 60% accuracy, while HostTracker achieves 90% accuracy for 95% of the ranges. The poor accuracy of the naive method suggests that it is highly likely for a host to have multiple user IDs, and not all IDs will be used each time a host connects to the Internet. Merely looking at user IDs will miss these host-IP bindings, while HostTracker can correctly identify these bindings and group those user IDs together.

For the cases where multiple hardware IDs were related with one host, initially we suspected that our grouping step might be over-aggressive in grouping IDs together. However, we manually checked many such cases and found even a static IP address with one user ID (i.e., without grouping) can sometimes have multiple hardware IDs, possibly behind one NAT host. Since our method does not perform more fine-grained classification for hosts behind NATs or proxies, these cases may be a significant source of inaccuracy in our tracking. We are further investigating these cases.

5.4 Mobility Analysis for Users

All our tracking so far is limited to within an IP range. However, users may travel and hence either move their hosts (e.g., laptops) or use different hosts (e.g., office desktop vs. home computers). Intuitively, a stable user usually has one or two IP ranges where they access the Internet frequently, but may occasionally travel to other locations. Thus although users may log in from many different network locations, we expect a majority of the normal user IDs will be classified as part of a tracked group at only one or two IP-address ranges.

As shown in Figure 9, over 80% of tracked IDs logged in from at least two different IP ranges. However, about 85.5% of tracked IDs were tracked at only one or two IP ranges. A relatively small fraction (12.5%) of the tracked IDs traveled frequently and were observed logging in from more than 5 different IP ranges.

We then compare the mobility of tracked users vs. untracked users. While it is completely plausible for a legitimate user to travel a lot and hence be associated with many different IP ranges, those untracked users who also logged in from a large number of IP ranges were highly suspicious. These transient and distributed user IDs are very characteristic of botnet attacks and we suspect them to be spamming email accounts signed up by botnet attackers [32].

To test our hypothesis, we examine all the highly mobile users that logged in from at least 10 different IP ranges. There exist

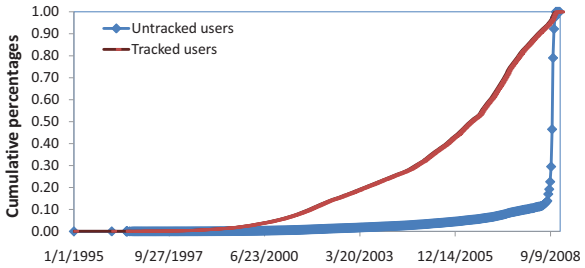
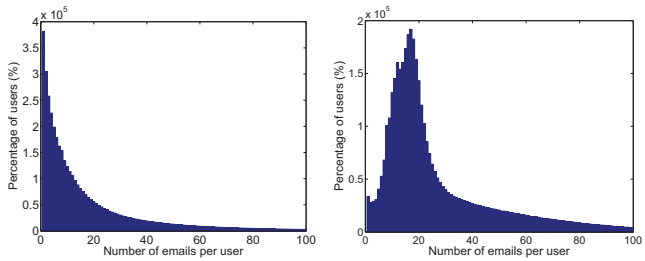


Figure 10: The cumulative distributions of the account signup date for the two different types of user IDs.

about 4.3 million such untracked IDs and 5 million such tracked IDs. Since botnet account-signup attacks are mostly recent (starting from around summer 2007), we compare both the account signup dates and their email-sending statistics for the two different populations. Note the two sets of users are selected using the same criterion of having logged in from at least 10 IP ranges. Further, since our host-tracking graph generation does not rely on either the user-account signup data or the email-sending data, our user classification (tracked vs. untracked) should not bias the two statistics we examine.

As shown in Figure 10, about 94% of the untracked IDs were signed up very recently since July 2008. For the set of tracked user IDs, their account signup dates were evenly distributed over time.

The email-sending histograms of the two different types of user IDs are strikingly different as well (see Figure 11). For the set of tracked IDs, despite the fact that the corresponding users logged in



(a) tracked users (b) untracked users

Figure 11: Histogram of the number of emails sent by tracked and untracked IDs who logged in from at least 10 IP ranges.

from a large number of IP-address ranges, the majority of them sent only 1 email throughout the entire month. However, for the set of untracked IDs, the number of emails sent per user shows a strong spike at around 20 emails per user. The strong correlations of their email-sending statistics were possibly due to coordinated botnet activities. Our study suggests that whether a user is associated with a tracked host or not can be a discriminating feature to identify botnet activities. In the next section, we systematically explore how we can exploit the host-tracking results for security applications.

6. APPLICATIONS

The host-tracking results provide rich information for network security. In this section, we consider several applications to demonstrate their power. First, we can exploit the host-tracking results for inferring the number of compromised machines and the sizes of botnets. Second, we can build normal-user profiles and statistics to help distinguish abnormal or malicious activity. Third, we can carry out forensic analysis and identify malicious activities not previously captured. Finally, we introduce *tracklists*, which are essentially blacklists of hosts (rather than of IP addresses).

Our study focuses on a recent large-scale botnet attack as our example throughout this section. In this attack, botnet hosts created tens of millions of user accounts from large free email service providers such as Yahoo!, Hotmail, and Gmail, and subsequently used them for spamming. Detecting these large quantities of malicious accounts has been challenging, as each account was used to send only a small number of emails to appear legitimate. Recent work [38] offers one solution to identify the set of accounts that were highly correlated with each other in their IP-address usage. We will augment the set of already detected accounts with the host-tracking information.

6.1 Estimation of Malicious-Host Population

It has been difficult to estimate the number of compromised hosts because of their dynamic IP-address assignment. In an extreme case, even if there is only one compromised host in a range, it can change IP addresses frequently, thus appearing to be many active hosts. As a result, previous work resorted to probabilistic models to estimate botnet size [39]. With host-IP bindings, we can now try to compute the number of malicious hosts more accurately.

Figure 12 shows the number of malicious hosts identified by HostTracker using the known malicious accounts. Each data point along the X-axis corresponds to a botnet campaign. We compare the number of IP addresses and the number of hosts. (The Y-axis is in logarithmic scale.) Not surprisingly, the number of IP addresses is significantly larger than the actual host population. For large botnets, the number of IP addresses can be two to three orders of magnitude bigger than the actual host number. This result suggests that some ranges are very dynamic and that hosts change IP addresses almost every day in our one-month trace. Therefore, using IP addresses to estimate botnet size can yield large variation. Moreover, even if we identify a malicious activity and block the IP address immediately, the blocking may not be effective even several hours later. We will further investigate the false-positive rate of IP-based blacklisting in Section 6.4.

The ASes with the largest number of malicious IP addresses and hosts also differ. We have examined the list of 10 ASes with the largest number of botnet hosts and the list of 10 ASes with the largest number of malicious IPs. Both lists include large ISPs across the globe. It is interesting that only four ASes appear in both lists and their ranks are dramatically different. Those ASes that appear in the first list but not in the second may have more dynamic IP-address assignment. Even a small number of compromised hosts

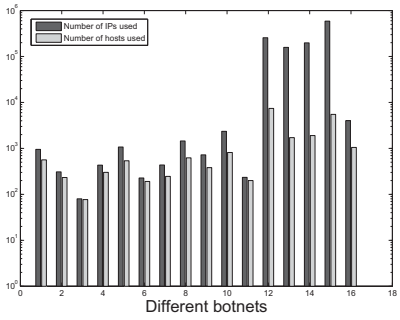


Figure 12: Botnet host size vs. botnet IP size.

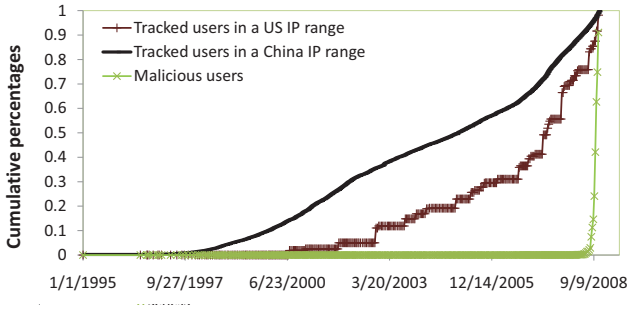


Figure 13: Account signup-time comparison between tracked user IDs (from two different IP ranges) and malicious user IDs.

may appear at many IP addresses. On the other hand, those ASes that appear in the second list but not in the first may have relatively more compromised hosts.

6.2 Building Profiles for Normal Users

In many applications, it is desirable to generate user statistics to help understand normal user behavior and distinguish abnormal activities. However, it is difficult to obtain such statistics as malicious activities can pollute the data, especially when attackers create millions of accounts.

The first question we explore is whether tracked users are normal users and thus can be used to generate statistics. To answer this question, we first calculate the intersection between the captured 220 million tracked users and the 5.6 million known malicious IDs derived from [38]. Only 50.2K (0.02% of the 220 million) are in this intersection. This is significantly lower than the percentage of known malicious IDs in the overall population (1%), suggesting that tracked users are highly likely to be normal users. Thus they can serve for building normal user profiles. The presence of a small percentage of malicious IDs will not affect the overall population statistics.

The profile of normal users can be dramatically different from that of malicious ones. Furthermore, the user profiles from different IP ranges can also be noticeably different. We look at two example features:

Feature 1: Signup time. Figure 13 shows that tracked IDs were signed up continuously over years, while malicious IDs were signed up during a very short time period, during which the sign-up activity was intense. Also, the signup times of tracked users at two different IP ranges are drastically different. The signup time in the US range is relatively early and with a long history, while the China users are relatively new.

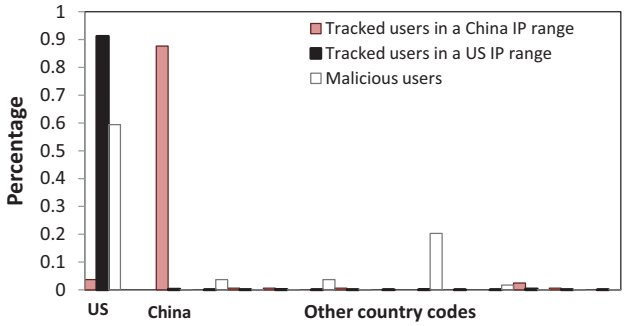


Figure 14: Country-code comparison between tracked user IDs (from two different IP ranges) and malicious user IDs.

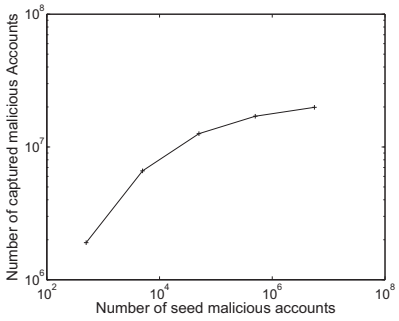


Figure 15: Correlations between the number of additionally captured IDs and the seed size.

Feature 2: Signup country code. When signing up, users will be asked to input their country code. Although some legitimate users may choose to enter a false country code, we expect them to be a minority. Figure 14 shows that the country code of the tracked IDs from US and China ranges differ significantly. Interestingly, a large number of malicious IDs choose US as country code. Although attackers can freely choose country codes when they sign up accounts, it is difficult for them to pick the correct ones since the accounts may be used all over the world.

User profiles can be leveraged to differentiate legitimate IDs and malicious IDs, and we use these profiles to help filter the false positives of newly discovered malicious IDs in the next subsections.

6.3 Postmortem Forensic Analysis

Using known malicious activities as a seed, we can conduct postmortem forensic analysis to identify more malicious activities that were not captured before. For this analysis, we use two types of inputs. The first is the set of 5.6 million previously detected malicious accounts. The second is the host-tracking graph derived by HostTracker, including information on the host-IP binding durations and the host types (tracked hosts and proxy hosts).

6.3.1 Forensics for Tracked Hosts

Suppose a malicious activity is discovered at time t at a tracked host. It indicates that the corresponding host is compromised. So we can use the host-tracking information to trace this host's past host-IP bindings from the current time back to t and mark the activities during this period as suspicious.

For certain applications, a host-IP binding window marked as suspicious can serve as a criterion for identifying malicious traffic. For example, if we suspect a host is compromised, we can treat all the port-25 mail relay traffic from this host as suspicious. However, for many other applications, this criterion is too coarse because there can be legitimate user activities on compromised machines. In this case, the ID-grouping information can help. In particular, activities of a tracked ID with a long history are more likely to be legitimate than activities of untracked IDs.

Relying on the ID-grouping information and the 5.6 million seed malicious IDs, we can identify an additional 9.4 million malicious IDs (167% of the input user ID size). To evaluate the false positive rate, we applied various criteria, including comparing with a whitelist of premium user accounts, checking their signup dates, verifying their naming patterns, and manual verification by sampling. The false positive rate is around 0.4%. We further examine many of these false positives, and find that most of them can be eliminated by applying the normal-user profiles.

6.3.2 Forensics for Proxies

Forensics is as important for proxies as for regular hosts. However, proxies must be treated differently. When malicious activity is detected from a proxy, we cannot blindly mark all its traffic as malicious since many legitimate users could use the same proxy. In this case, we employ the normal-user profiles to filter potential false positives. For every ID that binds to a suspicious proxy host, we perform tests using the signup-time and country-code features, and output accounts that significantly deviate from the normal profiles as suspicious. Using host-tracking information alone, we capture additional 12.4 million IDs that share the host-IP bindings with the seed accounts. Using the criteria of Section 6.3.1, we estimate the false positive to be 36.8%. After applying the profile-based filtering, we reduce the number of suspicious IDs to 6.4 million (3.2 million overlap with the ones identified in Section 6.3.1) with a false positive rate of 0.5%. Therefore using normal-user profiles can successfully eliminate the majority of the false positives.

6.3.3 Seed-Size Analysis

Since the seed malicious IDs can help identify more attack activities, we are interested in how many seed malicious IDs are necessary for this purpose. We study the correlations between the number of seed IDs and the number of additionally captured malicious IDs. Figure 15 shows the number of new malicious IDs that we can identify by varying the number of seed malicious IDs. Using just 500 malicious IDs as seed, we can capture over 1 million additional IDs. When the seed number grows, the ability to detect further malicious IDs grows as well, but with a slower growth rate. When using all 5.6 million known malicious IDs as seed, we can capture in total about 12.6 million additional malicious IDs (from both tracked hosts and proxy-like hosts) with an overall false positive rate of 0.4%. We plan to use these 12.6 additional malicious IDs as seed to capture even more malicious IDs.

We further examine the number of malicious IDs captured from different IP ranges. Figure 16 shows that, for a majority of the ranges, the numbers of additional malicious IDs identified are orders of magnitude larger than the seed size. For a few ranges, even one or two seed malicious IDs can help effectively identify hundreds or thousands of malicious IDs.

6.4 Real-time Tracklists

The forensic analysis results presented in the previous section can yield a postmortem host-aware blacklist. In this section, we show that the host-tracking information can be leveraged to build a real-time host-aware blacklist, which we call a tracklist. The main difference is that the tracklist does not have prior information about the host-IP bindings.

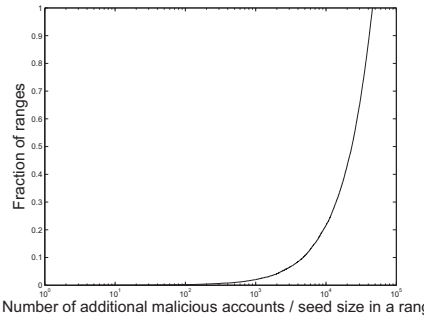


Figure 16: Cumulative distribution of the ratio between the number of additional captured IDs and the seed size across different IP ranges.

After we observe a malicious activity (as seed), we check whether the activity was associated with a past host-IP binding inferred by HostTracker. If so, we may deduce that the corresponding host is compromised. In this case, we can start blocking further malicious activities from this host at its current IP address. Otherwise, if the seed malicious activity was not associated with any host-IP bindings (because of the limited coverage of HostTracker), we start blocking the current IP address similar to traditional IP-blacklist based approaches.

To avoid over-aggressively blocking subsequent legitimate activities, we need to stop blocking this host under either of the following two conditions: (1) we observe the tracked ID group associated with the compromised host jumps to another IP address, (2) a different tracked ID group starts using this IP address. In the first case, we can follow the trail of the compromised host and block the new IP address.

Tracklists have several advantages over existing blacklists. They can follow the activities of the tracked hosts and hence enable us to block malicious activities despite changes of IP addresses. When blocking, we are still able to let the traffic generated by tracked IDs through, hence can reduce the false positives.

To quantify the effectiveness of the tracklist approach, we compare it with two existing IP-blacklist approaches. The first approach is to conservatively block each IP address for a short duration (one hour in our test) as soon as malicious activity is discovered. The second approach is to block the IP address infinitely. Table 3 shows that blocking infinitely results in a very large false positive rate (52.8%). Even when we shorten the blocking duration to one hour, the false positive rate is still 34.1%. Since malicious IDs share hosts with normal users, even blocking for one hour will reject legitimate requests. With tracklists, we can derive the tracked IDs from history and still allow their normal access. The false positive rate can thus be reduced to 4.9%. We investigate the false positives of tracklists, and find that most of them are associated with proxies. Using normal-user profiles combined with tracklists, we can further reduce the false positive rate to 0.1%.

	# of captured IDs	False positives rate
Block infinitely	44699757	52.8%
Block one hour	27940558	34.1%
Tracklist (infinitely)	28942360	15.6%
Tracklist (one hour)	16013895	4.9%
Tracklist (infinitely) with profiles	20813991	0.6%
Tracklist (one hour) with profiles	14272221	0.1%

Table 3: Comparison between the commonly used blacklists and tracklists.

7. DISCUSSION AND CONCLUSION

Even with access to a large email provider's log, we still have a limited view of the Internet. There are many users who do not use this email service and hence we cannot track their hosts. To increase coverage, one could leverage other forms of IDs, for example, social network IDs or various cookies. HostTracker can easily take in different types of logs and perform analysis on the union of them. In this respect and in others, there appears to be much scope for further improvements of our work, with increased tracking coverage and accuracy.

Our main reason to track hosts is to identify malicious activities. In this paper, we demonstrate how the host-tracking results can help identify malicious accounts. Those accounts are typically not tracked because they are not associated with fixed hosts. Binding malicious accounts to fixed hosts not only increases their risk of being detected, but also limits the attack traffic, since compromised hosts are not up all the time. The malicious accounts appearing at proxies may have a higher chance of evading detection by mimicking legitimate account profiles. But binding many malicious accounts to proxies also increases the chance of them being all detected and blocked, and proxy activities may be subject to stricter security tests. In addition to identifying malicious accounts, host-IP bindings can also be applied in other security settings, such as detecting and preventing DoS or scanning attacks.

For normal users, being trackable may sometimes have benefit. For instance, normal users could be notified when their computers are believed to be compromised. With such a service, users could potentially get their computers patched immediately, thus minimizing damage.

On the other hand, the use of application logs to derive host-tracking information raises privacy concerns. Users who do not wish to be tracked can adopt tools such as large HTTP proxies or anonymous routing [31]. In contrast, botnet attackers cannot easily evade HostTracker while retaining their attack effectiveness. The limited availability of anonymous networks makes them unsuitable for large-scale attacks.

It is sometimes said that the Internet is anonymous and hosts are not accountable. While the Internet lacks strong accountability, this lack does not mean that users and hosts are truly anonymous. In this paper, we show that when IP addresses are augmented with unreliable application IDs, many activities can be attributed to the responsible hosts despite the existence of dynamic IP addresses, proxies, and NATs. The host-IP binding information can be used to effectively identify and block malicious activities by host rather than by IP address. In the next-generation Internet, anonymity and traceability should be offered and reconciled by design rather than by accident.

Acknowledgements

We thank Úlfar Erlingsson, Michael Isard, Qifa Ke, Vijayan Prabhakaran, Ollie Williams, Ted Wobber, and our shepherd David Andersen for their valuable advice. We are grateful to Mike Bidgoli, Cindy Cai, Hersh Dangayach, Jeff Davis, Eliot Gillum, Geoff Hulten, Linda McColm, Ivan Osipkov, Krish Vitaldevara, and Jason Walter for providing us with data and feedback on the paper.

Martín Abadi is also affiliated with UCSC.

8. REFERENCES

[1] D. G. Andersen, H. Balakrishnan, N. Feamster, T. Koponen, D. Moon, and S. Shenker. Accountable Internet Protocol (AIP). In *Proc. of ACM SIGCOMM*, 2008.

[2] D. S. Anderson, C. Fleizach, S. Savage, and G. M. Voelker. Spamscatter: Characterizing Internet scam hosting infrastructure. In *Proc. of the 14th USENIX Security Symposium*, 2007.

[3] S. Bellovin, M. Leech, and T. Taylor. ICMP traceback messages. Internet draft, work in progress, 2001.

[4] Multi-NDSBL lookup. `http://www.completewhois.com/cgi2/rbl_lookup.cgi?query=148.202.33.219&display=webtable`, 2007.

[5] A. Blum, D. Song, and S. Venkataraman. Detection of interactive stepping stones: Algorithms and confidence bounds. In *Proc. of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2004.

[6] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *Proc. of USENIX LISA Systems Administration Conference*, 2000.

[7] M. Casado and M. J. Freedman. Peering through the shroud: The effect of edge opacity on IP-based client identification. In *Proc. of the 5th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2007.

[8] K. Chiang and L. Lloyd. A case study of the rustock rootkit and spam bot. In *Prof. of the First Workshop on Hot Topics in Understanding Botnets (HotBot)*, 2007.

[9] D. L. Donoho, A. G. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford-Chen. Multiscale stepping-stone detection: Detecting pairs of jittered interactive streams by exploiting maximum tolerable delay. In *Proc. of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2002.

[10] R. Droms. Dynamic host configuration protocol. *RFC 2131*, March 1997.

[11] Dynablock dynamic IP list. `http://www.spamhaus.org/pbl/index.lasso`, recently aquired by spamhaus, 2007.

[12] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. *RFC 2827*, May 2000.

[13] G. Gu, R. Perdisci, J. Zhang, and W. Lee. Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proc. of the 15th USENIX Security Symposium*, 2008.

[14] G. Gu, J. Zhang, and W. Lee. Botsniffer: Detecting botnet command and control channels in network traffic. In *Proc. of NDSS*, 2008.

[15] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling. Measurements and mitigation of peer-to-peer-based botnets: A case study on Storm worm. In *First USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[16] T. Killalea. Internet service provider security services and procedures. *IETF RFC 3013*, Nov 2000.

[17] T. Kohno, A. Broido, and K. Claffy. Remote physical device fingerprinting. In *Proc. of the IEEE Symposium on Security and Privacy*, 2005.

[18] B. W. Lampson. Computer security in the real world. *IEEE Computer*, 37(6):37–46, June 2004.

[19] J. Li, M. Sung, J. Xu, L. Li, and Q. Zhao. Large-scale IP traceback in high-speed Internet: Practical techniques and theoretical foundation. In *Proc. of the IEEE Symposium of Security and Privacy*, 2004.

[20] X. Liu, A. Li, X. Yang, and D. Wetherall. Passport: Secure and adoptable source authentication. In *Proc. of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2008.

[21] Nmap free security scanner. `http://www.insecure.org/nmap/`.

[22] Project details for p0f. `http://freshmeat.net/projects/p0f/`.

[23] RFC1661: The Point-to-Point Protocol. `http://tools.ietf.org/html/rfc1661`.

[24] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. In *Proc. of ACM Sigcomm*, 2006.

[25] A. Ramachandran, N. Feamster, and D. Dagon. Revealing botnet membership using DNSBL counter-intelligence. In *2nd Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, 2006.

[26] A. Ramachandran, N. Feamster, and S. Vempala. Filtering spam with behavioral blacklisting. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2007.

[27] Route views project. `http://www.routeviews.org`.

[28] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *Proc. of ACM SIGCOMM*, 2000.

[29] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer. Hash-based IP traceback. In *Proc. of ACM SIGCOMM*, 2001.

[30] Spamhaus policy block list (PBL). `http://www.spamhaus.org/pbl/`, Jan 2007.

[31] P. Syverson, D. Goldschlag, and M. Reed. Anonymous connections and onion routing. In *Proc. of the IEEE symposium on Security and Privacy*, 1997.

[32] Trojan now uses Hotmail, Gmail as spam hosts. http://news.bitdefender.com/NW544-en--Trojan-Now-Uses-Hotmail-Gmail-as-Spam-Hosts.html.

[33] X. Wang and D. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter-packet delays. In *Proc. of ACM Conference on Computer and Communications Security (CCS)*, 2003.

[34] Whois.net–domain research tools. `http://www.whois.net/`.

[35] Y. Xie, F. Yu, K. Achan, E. Gillum, M. Goldszmidt, and T. Wobber. How dynamic are IP addresses. In *Proc. of ACM SIGCOMM*, 2007.

[36] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming botnets: Signatures and characteristics. In *Proc. of ACM SIGCOMM*, 2008.

[37] Y. Zhang and V. Paxson. Detecting stepping stones. In *Proc. of the 9th USENIX Security Symposium*, 2001.

[38] Y. Zhao, Y. Xie, F. Yu, Q. Ke, Y. Yu, Y. Chen, and E. Gillum. Botgraph: Large scale spamming botnet detection. In *Proc. of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[39] L. Zhuang, J. Dunagan, D. R. Simon, H. J. Wang, and J. D. Tygar. Characterizing botnets from email spam records. In *Proc. of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.