Critter: Content-Rich Traffic Trace Repository

Vinod Sharma
USC/ISI
vinodsha@usc.edu

Genevieve Bartlett
USC/ISI
bartlett@isi.edu

Jelena Mirkovic
USC/ISI
mirkovic@isi.edu

ABSTRACT

Access to current application and network data is vital to cybersecurity and networking research. Intrusion detection, steganography, traffic camouflaging, traffic classification and modeling all benefit from real-world data. Such data provides training, testing, and evaluation as well as furthers efforts to reach ground truth.

Currently available network data—especially data with application-level information—is often outdated and is either private or customized to specific, narrow research needs. The biggest hurdle to obtaining such content-rich data is addressing the huge privacy risks associated with sharing such complex and open-ended data. In this paper we present a data sharing system called Critter-at-Home which addresses these challenges. Critter connects end-users willing to share data with researchers and strikes a balance between privacy risks for a data contributor and utility for a researcher.

1. INTRODUCTION

Networking and cybersecurity research critically need publicly available, fresh and diverse *application-level* data, for data mining and for validation. For example, spam filtering, intrusion detection and traffic classification all require application data to learn trends in legitimate network use, to establish ground truth, and for realistic evaluation of proposed systems. Despite these needs, there are very few publicly available network traces which contain application-level data, The few datasets publicly available suffer from being outdated, and from containing very specific data useful only to some researchers.

The main reason for this lack of content-rich network data is the enormous privacy risk that sharing such data creates. The networking community has long worked to solve a simpler problem—mitigating the risks associated with sharing only content-*less* network data—with little progress. Sharing application-level data greatly increases privacy risks, because such data is rich with personal and private information, such as human names, social security numbers, phone numbers, usernames, passwords, credit card numbers, etc. that Internet criminals can monetize. These risks increase with the longevity of the data, as demonstrated by cases where attackers were able to compromise privacy after a dataset release either through direct attacks on the dataset [1] or through correlation of one or more datasets [2, 3].

In this paper we look at the pressing needs for content-rich data, and introduce Critter-at-home—a live content-rich repository which addresses the immense privacy risks associated with sharing such data.

Critter-at-home connects end-users willing to share data—called *data contributors*—with researchers and aims to strike a balance between privacy risks for the contributor and utility for the researcher. Critter protects data contributors' privacy and control over their contributed data in several ways:

1. No human ever looks at the raw data collected by the Critter system. Instead, researchers query the data through a portal and receive aggregated responses to these queries—such as counts or distributions. Critter applies k-anonymity to maintain privacy and prevent researchers asking queries which could identify an individual.

2. All data collected is encrypted via public-key before being stored.

3. Lastly, a contributor can withdraw her data from Critter at any time. Data can be stored locally on a contributor's machine, or optionally remotely. In either case, each contributor maintains control over her data. This differs substantially from traditional network traces and other datasets, where once the data has been collected, the longevity of the data is determined by the dataset owner, not by the contributers.

In this paper we discuss the need for content-rich data (Sec. 2.1), the challenges associated with collecting such data (Sec. 2.2), and how we answer these challenges in the Critter-at-Home design (Sec. 3). We present several demonstrations of queries and results using our initial deployment (Sec. 4).

2. THE CASE FOR CONTENT-RICH DATA

We start by first exploring the needs of the network and systems communities for content-rich data, and then address why such data is not currently publicly or widely available. These needs and challenges will define the goals Critter-at-Home must aim for in order to provide utility.

2.1 Needs

To understand researcher needs for passive data, we survey work published in Sigcomm 2013 and IMC 2013. We look for works which use passive network data and explore where this data was collected, if the data is made publicly available, what network-level data was required, and which platform(s) data was collected on such as an end-user machine, a middle point—such as an exchange point

WISCS'14, November 3, 2014, Scottsdale, Arizona, USA.
Copyright 2014 ACM 978-1-4503-3151-7/14/11 ...$15.00.
http://dx.doi.org/10.1145/2663876.2663886.

	Content			Perspective				Platform				
Conference	-rich	-less	both	client/peer	server	both	network	PC	cellular	other	Total	Can use Critter
Sigcomm 2013	3	3	1	4	1	2	0	5	2	0	7	4
IMC 2013	2	6	1	4	1	1	3	6	2	3	9	3

Table 1: A summary and break-down of papers which used passive data in their work.

or router—or at a server. A break down of this information is in Table 1.

We first note that of the 16 works which use passively collected data, only one dataset is publicly, or semi-publicly available: the Ono BitTorrent dataset. Ono relies on aggregation and anonymization to protect contributor information and does not include application-level content beyond statistics on BitTorrent usage. Quite possibly other datasets could be made available through working through legal agreements, but we note that this process is time consuming and to some extent discourages reuse of datasets. The rest of the 15 efforts using passive data use privately collected data, collected for the specific work at hand. This custom collection ensures the data meets the researcher needs, but we note that this data comes from only one, or in rare cases two, environments the researchers had access to.

Of the 16 papers we surveyed, 7—roughly half—used application-level data—e.g. content of DNS responses. Many used both data from a client and server perspective, but we note a large number of works using data from the client-perspective: 11 used data collected at a client or peer end-user machine.

In all works, the oldest dataset used was collected four years prior to the published results. The majority of works used data collected from the Internet, but a significant number (4) looked at passively collected data from cellular networks.

Based on the usage, we claim there is a need for fresher, and more diverse content-rich (and content-less) data, from both a client (or peer) and server perspective, and from a wide variety of devices, including cell phones. Currently such data, especially content-rich data is not publicly accessible, and comes from only a handful of environments. In the next section, we address why such data is not currently readily available, and outline what challenges must be overcome to provide such data.

2.2 Challenges

There are three main challenges in collecting content-rich data. First, to ensure we get diverse data, we need to motivate data providers to share such data. Second, we need to address the massive privacy risks associated with sharing content-rich data. Last, we need to understand how to process and store this data to enable research utility. These three main challenges are intertwined: Appropriately addressing the privacy risks can alleviate the concerns data providers have with sharing content-rich data; however, there is a trade-off between privacy and research utility.

Motivating data providers to share data *at the very least* requires any sharing be low-effort and have limited risk and cost to the sharer. This means any system for collecting data must be cross-platform, easy to install, consume very few system resources and be secure.

Storing and processing content-rich data is more complex than simple network packet headers. Unlike network and transport protocols, there are a near limitless number of applications and settings which contribute to content-rich data. This open-endedness means content-rich data does not lend itself well to a rigid database schema and more resources are needed to process such data compared to well-defined data. If we reduce the complexity of the data in some manner in order to speed up processing or reduce storage, we risk limiting the research utility of the data.

Addressing privacy risk poses the biggest set of hurdles to sharing content-rich data. Packet payloads include an unbounded amount of private information—such as addresses, personal conversations, bank information, social security numbers, and so forth. To compound the problem, some private information may not appear to pose a risk—such as product and location preferences—but may still be used by someone acquainted with the data contributor to uniquely identify him or her. Many efforts to anonymize data with such unbounded content have resulted in unintended information leaks. Some examples include: (1) the de-anonymization of the Netflix movie ratings dataset when correlated with the Internet Move Database (IMDb) [2] (2) privacy violations by the public release of AOL search data [1] and (3) the de-anonymization of medical records by correlating them with a publicly available voter database [3].

Standard methods for protecting publicly available network data through sanitization and anonymization are not well-suited to protect content-rich network data. Currently, publicly available network data from Internet users is *sanitized*—a process which removes most or all of the application-level data and anonymizes sensitive information such as IP and MAC addresses. Despite such mitigating measures, sanitized data is still highly vulnerable to both active and passive de-anonymizing attacks [4, 5, 6, 7]. Additionally, sanitization offers poor protection against future attacks. Once a user has downloaded the data, the provider has no control over how data is used, and there are no mechanisms for a data provider to retract published data once an attack has been discovered. Due to the greater privacy risks that come with sharing content-rich data, building upon traditional sanitization methods to protect privacy is not an option.

A data provider can maintain better control over access to their shared data by storing shared data locally, and only allowing certain queries on the data. While this reduces the privacy risks associated with releasing data in its entirety, this also requires the data provider to donate disk space and CPU processing time. If the CPU and disk space requirements are too high, or unchecked, data providers will likely be less willing to share.

Ultimately, the biggest challenge is picking the right trade-offs which limit costs and risks to the data providers while providing utility to researchers. In the next sections, we discuss how the Critter Framework picks these trade-offs.

3. Critter-at-Home FRAMEWORK

Given the needs outlined in Section 2.1 for content-rich data and the challenges for obtaining this data in Section 2.2, we have the following design-goals:

1. **Data is fresh, diverse and content-rich.** Network service providers have typically been a great source for passive data (eg. [8, 9]), but these providers are not in a position to publicly offer continual, content-rich data given the privacy risks, storage costs, user dissatisfaction and liability associated with sharing such data. Instead of reaching out to service providers, Critter reaches out to individual users directly. Connecting to these individual *data contributors* directly, and providing mechanisms for these individuals to contribute continuously, ensures up-to-date data from a diverse set of environments.

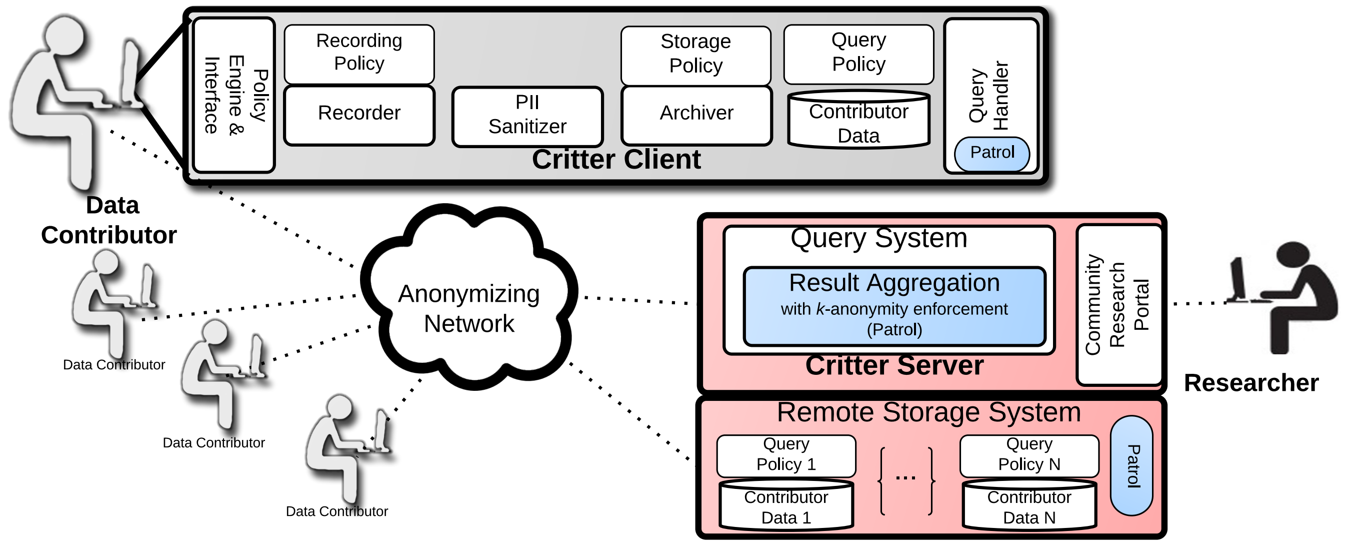

Figure 1: The basic overview of Critter-at-Home.

2. **Contributor identities and any personally identifying information (PII) are fully protected.** Private data such as Social Security Numbers, phone numbers and addresses can be stolen and monetized by criminals. One of our biggest goals is to ensure that a data contributor's identity and PII are never at risk due to the Critter system. We store the raw data in an encrypted format to prevent any unauthorized access. We further protect data contributors through a secure query framework called Patrol which only releases aggregate and prevalent results based on work done by J. Mirkovic [10]. Thus the privacy of a data contributor is further protected by using a "hiding in a crowd" approach.
3. **Contributors maintain full and flexible control over their privacy risks and data.** Critter gives contributors full control over their data and its usage at a fine-grain level including mechanisms to withdraw data fully at any point, store data remotely or locally, and contribute only what they are comfortable with.
4. **Sharing is easy.** We wish to make sharing easy to encourage end-users to participate in Critter. This includes an easy installation, and data collection which uses minimal resources.
5. **Data utility is maximized.** We minimally process data and store data in an almost "raw" format. This allows researchers to query data for features that interest them, rather than restrict queries to a predefined set of features extracted from the raw data.

In the following sections we discuss the Critter's threat model, and how Critter's design guards against these threats and protects data contributors, while maintaining high data utility. We will discuss the components of Critter and the specifics of collecting and querying data and anonymizing participation. Though we do not have the full design of Critter-at-Home implemented, we present our planned framework in its entirety, and note what has been implemented so far in our first version of Critter-at-Home.

3.1 Threat Model

The biggest challenge Critter-at-Home addresses is the privacy risks data contributors face. Specifically, there are three main threats to privacy:

1. **Data Stealing.** Contributor data may be stolen by a third party, not necessarily related to Critter-at-Home, e.g., a Trojan. The raw version of contributor data contains sensitive and private information that would not otherwise exist in one place. Data could be stolen from disk storage, from memory or from the network.
2. **Data Query Correlation.** Contributor data may be queried by someone familiar with the contributor, or someone with auxiliary information from other sources, for the purpose of linking query results to a specific contributor. For example, if query results exposed information about a contributor's salary range and employer, someone familiar with a contributor's employer would have an easier time guessing a contributor's salary.
3. **Contribution Correlation.** An observer may attempt to correlate a contributor's network behavior—such as IP address or a pattern of connections—to responses to queries and thereby private information about a specific contributor. Simply knowing a contributor is participating in sharing data increases the chance that a contributor's identity is linked to released information. Additionally, if an attacker can use observed network traffic and timing information to learn which queries a contributor responds to and which ones she does not respond to, the attacker can learn how results change with and without the contributor's response and have an easier time guessing information about a contributor.

In the following sections we will refer to these three threats—data stealing, data query correlation and contribution correlation as we discuss how Critter's design guards against these.

3.2 Architecture

The Critter-at-Homeframework consists of three main parts—a *Critter client* which runs locally on a data contributor's machine, a *Critter server* which collects and disseminates researcher's queries and replies and ultimately will also include a *Remote Storage System* which houses data for contributors who do not wish to store their data locally. Currently, we have not implemented the Remote Storage System and so all data is stored locally. Figure 1 depicts these components for Critter-at-Home and how they interact.

Data contributors (shown in gray in Figure 1) run the Critter client program on their local machines. The Critter client collects the contributor data via its Recorder module, according to the Recording Policy. In future versions, the client will pass data through a PII Sanitizer module to remove Personally Identifiable Information (PII), before encrypting the data with the contributor-generated symmetric key. Both PII-sanitization and encryption are done to address the **data stealing** threat in our threat model. Collected data is handled by the Archiver module which in future versions will allow for data to be stored remotely, and not just locally. In these future versions, this decision will be recorded in the Storage Policy and honored by the Archiver.

The Query Handler module in the Critter client polls the Critter server for new queries, whenever the contributor's machine is connected to the Critter network. The Handler decrypts data, and processes the query if it is permitted by the Query Policy. Currently, this Query Policy is rudimentary, but in future versions we plan to expand the control contributors have over the queries they wish to participate in.

In future versions to guard against **contribution correlation** the Query Handler will encrypt results and a portion of the query policy—which must be resolved centrally during aggregation—using the Critter server's public key. The process is similar for data stored on the Critter server, except the need for polling to obtain new queries is eliminated, and the data is encrypted by the server-generated symmetric key. To further protect against **contribution correlation**, in future versions contributors will connect to the Critter server via an anonymizing network, such as the Tor network [11, 12].

Researchers submit queries and obtain results via a community portal, which passes these queries on to the Query System, located on the Critter server. Queries submitted to the Critter server through the public portal, are stored, and handed out to contributors whenever they join the Critter network. Researchers can query on any features which interest them but they only receive aggregate responses (counts, histograms, etc.) to address **data query correlation**. These responses are synthesized from contributors' individual responses, after applying their Query Policies to ensure "hiding in the crowd" (Section 3.4.1). Aggregate responses may be finalized after a configurable time-frame or after predefined response-count is reached, or a query may be indefinite, and periodically all responses to it are aggregated and returned to the researcher.

3.3 Collecting Data

In this section we discuss how we collect and store data from data contributors. Figure 2 depicts that basic process for data collection and storage. In our current implementation, data is recorded and stored locally. Though all storage is local now, in future iterations we plan on allowing remote storage and anticipate some contributors will prefer this option over donating disk space to Critter-at-Home.

3.3.1 Recording Data

The data collection process takes place locally on a data contributor's machine. Since one of Critter's goals is easy sharing, the program needs to be cheap to run resource-wise so as not to impact a contributor's regular computer use. The client also needs to be portable between environments to encourage a diverse set of contributors to participate.

Currently we support Windows, Linux and Mac OS X, but we plan to explore expanding to smart phones and other devices. Contributors can limit the memory and disk usage of the Critter Client to tune and lessen the impact of running the client. When the disk space limit is reached, older recorded data is overwritten by newer data.

Application-level data could be recorded at the network layer, or by instrumenting individual applications. Currently we have chosen to collect data at the application-level using a local SOCKS 5 [13] proxy. While collecting at the network-level offers more data for analysis, we feel users may be uncomfortable with a "network sniffer" collecting their data. We believe users will feel more in control with our local proxy set up since the user actively directs traffic to the proxy. In the future, we plan to investigate user viewpoints on collection through questionnaires.

For our first iteration of Critter, we collect only HTTP traffic. We choose HTTP since web traffic is a major contributor to Internet traffic today [14], but we exclude recording HTTPS since this data is more likely to contain private data. As discussed, one of Critter-at-Home's design goals is to allow full and flexible control over any data collected. To this end, users can choose which sites they wish not to record traffic for, and users have a basic user interface to stop and start data recording with a simple button click. For example, contributors will be able to permit or prohibit recording at the application level ("never record traffic from BitTorrent"), content level ("never record movies I watch over HTTP"), domain level ("never record traffic to/from `www.reddit.com`"), physical location level ("never record when my laptop is on my home network"), or message level ("never record HTTP POST traffic").

In the future we plan to expand beyond HTTP. There are a large and growing number of popular applications (web-browsers, email-clients, instant-messengers) which support the SOCKS v5 protocol, which makes expanding to these applications trivial. To support any new applications, we also need to write parsers for the new application's data. This will be an iterative process as Critter-at-Home grows.

3.4 Querying Data

Figure 3 illustrates Critter-at-Home's querying process. The Query System implements a store and poll mechanism for queries. Queries are pulled by contributors—rather than pushed to contributors—because contributors may only connect to the Critter-at-Home network intermittently, even if they are running the data recorder on a continual basis. By storing queries and collecting responses over a period of time we can maximize the number of contributors responding to a query. Though the current implementation of Critter-at-Home does not handle complex query features, we anticipate supporting features such as rolling aggregation and ongoing queries which return periodic results.

In the future, to fully guard against the **contributor correlation** threat, query polls, queries and responses will all go through an anonymizing network. Though routing queries and responses through onion routing or a mix network may increase latency of query polling and responses, this latency will not be experienced by contributors or researchers. Response gathering is done over an extended period of time to maximize participation and query polling and response is handled in the background. When we implement the Remote Storage System, data housed on this system can be queried on a more instantaneous basis, but will otherwise be treated the same as any locally stored data.

3.4.1 Result Aggregation

To protect from **data query correlation** we aggregate responses from clients through our Secure Query System. When sharing privacy-sensitive data via Critter-at-Home the original data always remains under the control of its owner. The data owner releases information through responding to queries with a numerical value. These responses are aggregated on the Critter Server before being returned to a researcher. Since we release only aggregate responses,

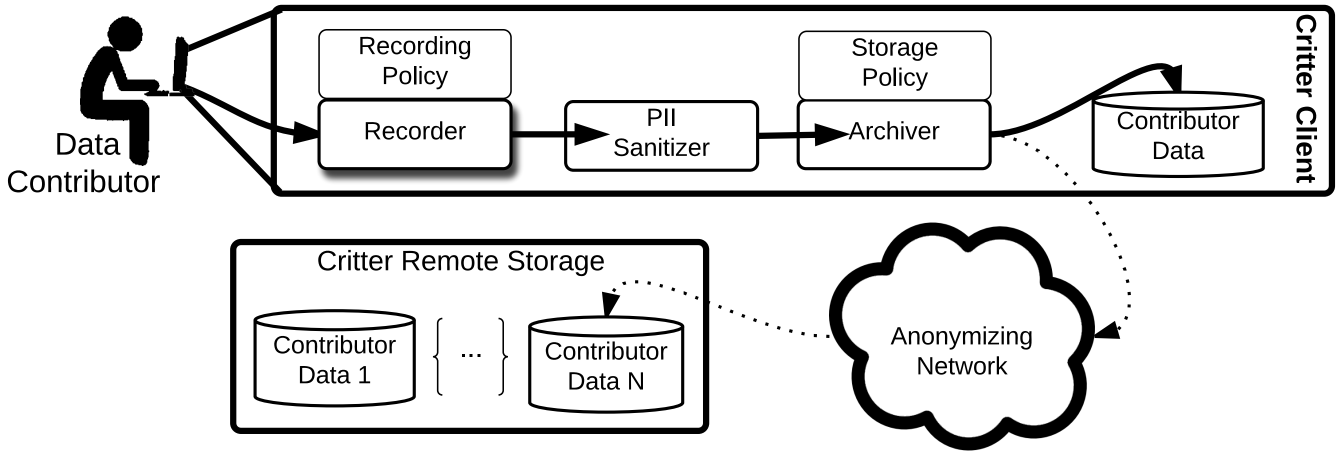

Figure 2: Data collection and storage in Critter-at-Home.

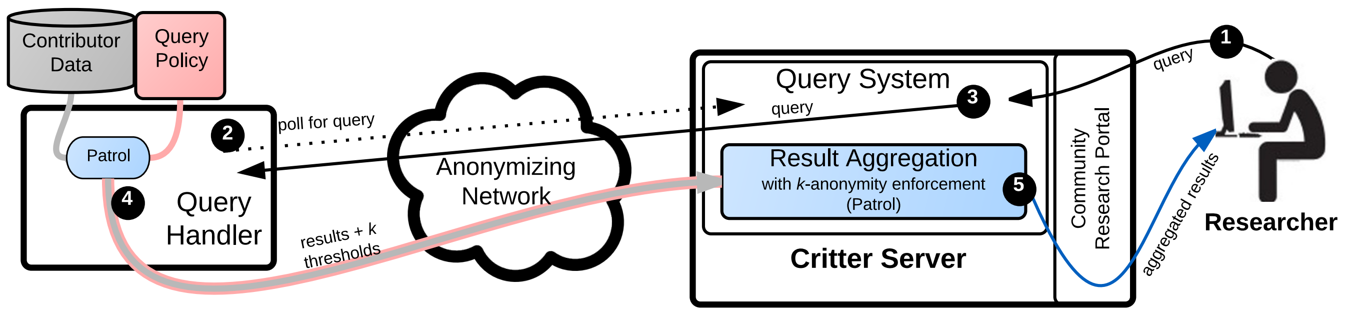

Figure 3: An overview of the querying process in five steps:(1) A researcher submits a query via the public portal. (2) Critter clients connect and poll for new queries via an anonymizing network. (3) The researcher's stored query is sent to clients. (4) Patrol processes the query if the Query Policy permits, and returns encrypted results along with information on how a contributor wants its response aggregated. (5) Aggregated results are stored by the query system and can be retrieved by the researcher. Patrol components (shown in light blue) will come from PI Mirkovic's NSF-funded project on secure queries.

many active and passive attacks that work against data sets such as sanitized network traces or sanitized logs are ineffective in our context.

We take a "hiding in a crowd" approach for privacy protection by enforcing k-anonymity [15] criteria before any result is returned to the researcher. If a researcher asks for how often users visit a particular website on a specific day, k-anonymity ensures that the returned result is a set of grouped responses such that each group has a single aggregated value representing at least k different contributors' replies. For example, if $k = 5$, the result might be: "5 users visited the website around 5 times and 6 users visited the website around 10 times", but the result would never be: "1 user visited the website 2 times per day" (since k-anonymity would not be met with $k = 1$).

We determine the membership of each group of counts by first ordering values and grouping responses with the same value together. If the number of responses for a value is less than k we either drop this value from the response or merge this group of responses with the adjacent group. We then check this merged group to see if its membership is at least k. If it is not, we repeat the process, otherwise we continue on with the next group. For now, the value associated with a merged group is the maximum value in that group, however we plan to investigate other methods of value assignment.

For our initial deployment we have fixed k at the low values of $k = 2$. In practice, this does not provide strong privacy protection. In the future, a contributor will be able to customize this value for their data, and researchers will be able to request desired ranges for k. Thus, our system will either honor the highest k value requested by a contributor or leave this contributor—and possibly others—out of an aggregated response when a researcher has requested a lower k value.

Understanding the freshness of data when doing result aggregation is important since older data may skew results form new data and vise versa. The age of the data in the Critter repository will vary from contributor to contributor, since the Critter client overwrites older data with new data when disk space limits are reached. Additionally, contributors who chose to store data remotely, may also stop contributing entirely leaving data to age. To address this variation in data age, queries are by default pulled from only the last week of data. Researchers can optionally specify a range of dates for the age of data included in a result.

3.5 Organizing Data

A natural and powerful way to search and store data is via a database with a fixed schema. Databases allow for efficient searches and there are a large number of database tools available for organizing and querying. Unfortunately, such a database schema is difficult to predefine given the complexity of the data Critter-at-Home collects. Though we focus on only HTTP data currently, as we add support for collecting content from other applications, this data complexity will only increase.

To deal with this complexity, we use a very simple database schema with only a few fields: the date and time an HTTP TCP flow began and ended, a breakdown of HTTP headers for each message in the flow, and the content of each message. This simple schema, however, does not enable complex queries. To address research utility, we allow restricted regular expression matching on fields such that only numerical results from a regular expression are allowed. This expression matching enables a wide-range of questions while not requiring us to understand and organize collected data *a priori*.

4. DEMONSTRATIONS

While we have not completed our work on Critter-at-Home, we have a working prototype of key components and currently we have a small base of six beta-test contributors. Here we discuss the specifics of our current implementation and present a few demonstrations of what can be asked and answered through Critter-at-Home.

4.1 Current Implementation and Data

Our initial implementation does not fully realize the complete Critter-at-Home plan. We have a Query System with a Result Aggregator and a basic Research Portal. The clients have a rudimentary query and recording policies in place, but we do not have a Remote Storage System so clients store data locally. Clients are set to store 10GB of data, and when this limit is hit, the client writes over older records with newer records.

We currently focus on HTTP traffic only. The Critter client is implemented in Python and collects application-level (HTTP content and headers) through a SOCKS 5 proxy [13] run from within the Critter client program. Contributors point their web browsers at this proxy and the Critter client handles storage, as well as polling and running queries over the data.

The Critter Query Server, which handles storing queries and responding to clients looking for queries is also implemented in Python. The research portal, where queries are inputted into the system, is via a PHP web interface.

To support multiple platforms, we implemented the Critter Client in Python and using PyInstaller [16] we have created binaries for Linux, MacOS X and Windows. Each binary has a platform specific installer, so installation is straight forward and simple—an important step to encouraging data sharing.

Since we use a SOCKS proxy to record data, we have HTML headers and content, and have very limited network and transport information. In our database, each TCP connection is a record, with one or more request/responses associate with it. Currently, we do not link related TCP connections with the original web page request which triggered the connections. For example, if a browser uses separate TCP connections to download images for a web page, these images are not tagged or associated with the web page they belong to. This currently limits the types of queries we can ask, and we plan to address this limitation in future iterations.

In the following demonstrations, we draw from two separate Critter repositories. The first is a repository of dummy users (Virtual Machines that run a Critter client and browse the web as instructed). The second is the repository with our live test users.

4.2 Demo: Counting Calories

For our first demonstration, we look at a hypothetical question which needs HTML content to answer: If a cooking recipe site, such as myrecipes.com, advertises lower calorie recipes, are people more inclined to check out these recipes?

Since this is a demonstration of Critter-at-Home and not an exploration into people's eating habits, we create dummy data using four Virtual Machines (VMs) running the Critter client and skew the data to show that a hypothetical ad campaign for the lower calorie recipes does indeed affect user browsing habits. We first have our VMs browse more high calorie recipes—simulating users who visit the site before seeing advertisements for lower calorie recipes—and then after a set date of June 1st, we have the VMs browse more low calorie recipes—simulating users who had been exposed to a hypothetical ad campaign starting June 1st.

	Before Ad		After Ad	
	<300 cal	>300 cal	<300 cal	>300cal
VM 1	2	4	7	4
VM 2	1	1	6	1
VM 3	5	8	14	2
VM 4	1	12	15	0

Table 2: Raw (non-aggregated) responses to queries on calorie counts before and after an ad campaign for lower calorie recipes.

HTTP responses from myrecipes.com have a meta element (`<span itemprop="calories">`) which gives the calories per serving of a recipe. We use this to define a regular expression to count the number of recipes from myrecipes.com a VM "views" that have less than 300 calories (low cal) and how many have more than 300 calories (high cal). We run these two queries twice—once specifying that we want the view count for before June 1st, and once specifying that we want the view counts after June 1st (after the supposed ad campaign began).

Table 2 shows the non-aggregated raw responses to these queries run on dummy data from our VMs running the Critter client. The aggregated responses to the four queries with $k = 2$ are: (1) Before June 1st 2 users viewed 1 recipe and 2 users viewed around 5 recipes with < 300 calories. (2) Before June 1st 2 users viewed around 4 recipes and 2 users viewed around 12 recipes with > 300 calories. (3) After June 1st 2 users viewed around 7 recipes and 2 users viewed around 15 recipes with < 300 calories. (4) After June 1st 2 users viewed around 1 recipe and 2 users viewed around 4 recipes with > 300 calories. From these returned responses, a researcher could conclude the ad campaign was effective.

4.3 Demo: Content Type

In our second demonstration, we use the data collected from our beta-test users and look at the content type over HTML TCP connections. Such information could be useful for modeling HTTP traffic or for understanding how to tune traffic camouflaging for covert communication.

We target the content type in the request field of our database, and ask for the percent of (HTML) TCP connections which are of a specific type—plain text, HTML, javascript, image, audio, video and other (any content type not matching the other six types). This requires us to ask seven queries to our system, one for each content type.

Figure 4 shows the aggregated results across all clients. The Critter-at-Home aggregation system for six of our queries returned three aggregated groups and for the audio query—since many values were the same—only two groups were returned. Figure 4 shows the group values returned, labeled as "Min", "Mid" and "Max".

Not surprisingly, image content appears in a significant portion (14–24%) of a user's HTML connections. However, the majority of content type in this analysis appears to be "Other"—in other words, the regular expression used did not match any of the defined content types we queried on. This highlights the difficulty with working with regular expressions and restricting responses to numeric values. While regular expressions allow for very flexible queries, and numeric responses are easily aggregated for privacy protection, we cannot tell immediately what comprises this "Other" category. In

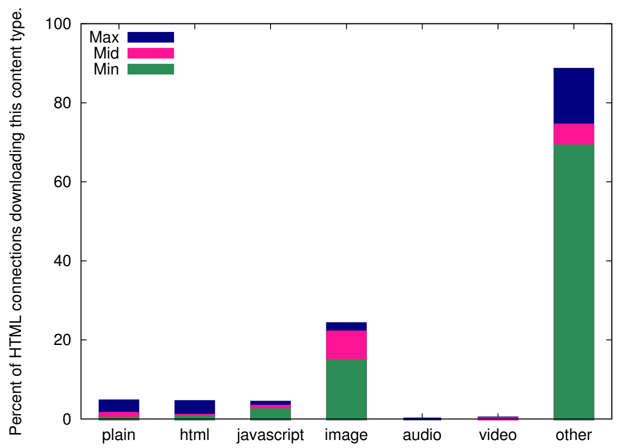

Figure 4: Min and max percent of HTML TCP connections which download a specific content type.

the future, we plan on exploring how short text responses can be protected and aggregated.

5. RELATED WORK

There are two areas of work related to Critter-at-Home: (1) work which addresses privacy risks inherent in data sharing and (2) frameworks to connect researchers to data sources.

There are a multitude of endeavors to overcome the privacy risks inherent in sharing Internet user data motivated by the great need for this data in research. Much work has been done on network trace sanitization, e.g. [4, 17], however as discussed in Section 2.2, these methods are not well suited to protect content-rich data and we do not discuss them further.

Pang and Paxson investigate the problem of parsing and sanitizing content-rich network data [18]. Their sanitization tool rewrites packets according to human-input policy scripts and can replace application-level header and content. While the packet transformation is flexible, human input process is tedious and error prone. Further, in presence of auxiliary data *no packet field is privacy-risk free*, thus sanitization in itself cannot be the entire answer to privacy risk in content-rich data sharing.

Another approach to releasing content-rich data safely, is to generate synthetic traffic based on some features drawn from the real data. The DARPA Datasets for Intrusion Detection System Evaluation [19, 20] are an example of such generated traffic. The original traffic was collected at an Air Force base, and the features that were mined from the traffic and replicated in synthetic traffic, as well as the synthesis approach for other traffic features, were not publicly disclosed. Researchers thus cannot quantify how data generation artifacts may influence their research outcomes. Another problem lies in the fixed choice of relevant traffic features to be mined and replicated. Researchers who are interested in a different set of features cannot utilize these datasets. In spite of these and other deficiencies [21, 22, 23], these datasets—generated in 1998, 1999 and 2000 by MIT Lincoln Laboratory—are still in active use today, over a decade after their release. This highlights the grave need researchers have for content-rich datasets.

There are a large and growing number of proposed approaches to support gathering and aggregating data from anonymous users in a privacy-preserving way, e.g. [24, 25, 26, 10]. While our approach to protecting privacy is based on work done by J. Mirkovic [10], Critter-at-Home differs from these efforts in that we take a holistic approach and address the specific challenges of storing, organizing, and protecting *unbounded* content-rich network data.

Frameworks for connecting researchers to data vary in complexity from portals providing curated data [27, 28, 29, 9, 30, 31] to systems for the policy process of sharing data [32], to distributed infrastructures for network monitoring [33, 34]. Our approach differs from these because our main goal is to provide a framework to collect data directly from individuals, and not from network administrators. This is a common goal for commercial enterprises for doing market research and quality of service studies. We are aware of only one other research endeavor which, like us, reaches out to a broad audience and creates an ongoing study. The BISmark project [35] offers users a free device which collects active measurements of link performance from a user's home network. We differ from BISmark on our measurement goal to passively collect information about network data—and specifically packet content.

6. DISCUSSION

We see a pressing need for publicly available, fresh and diverse content-rich data which currently is not being met. There are a large number of challenges to address with collecting such data: motivating users to share, storing and processing such unbounded data, and of course addressing the large privacy risks associated with sharing content-rich data.

We have designed a complete system which addresses these challenges and connects researchers directly to end-users willing to share information. We have a lot of work ahead of us, and many questions will need to be investigated as Critter-at-Home grows in both contributors and in research queries: What types of self-selection bias will we see with volunteer contributors and how do we mitigate and understand this bias? How does aggregation affect data utility? How does the intermittent nature of clients in the Critter-at-Home network affect results? How do we improve research utility while still protecting privacy? Can we extend Critter to running on other platforms such as smart phones?

We anticipate that end-users are interested in participating in research, and with an easy to install, low-risk system which gives users full control over their data, we expect users are quite willing to donate information. For contributors though, Critter-at-Home is not just about advancing research, but also about learning more about their systems and Internet behavior. Wide participation in projects like Panopticlick [36]—which give information on how unique and trackable a user's browser is—indicate that there is a broad audience who is interested in learning how their Internet behavior compares to others.

7. REFERENCES

[1] S. Hansell, "AOL removes search data on vast group of web users.," New York Times, August 2006.

[2] A. Narayanan and V. Shmatikov, "Robust De-anonymization of Large Sparse Datasets," in Proceedings of IEEE Security and Privacy 2008, pp. "111–125", IEEE, May 2008.

[3] L. Sweeney, "Weaving technology and policy together to maintain confidentiality," in "Journal of Law, Medicine and Ethics", vol. 25, pp. 98–110, 1997.

[4] J. Xu, J. Fan, M. H. Ammar, , and S. B. Moon, "Prefix-Preserving IP Address Anonymization: Measurement-Based Security Evaluation and a New Cryptography-Based Scheme," in Proceedings of the IEEE International Conference on Network Protocols, 2002.

[5] Q. Sun, D. R. Simon, Y. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical Identification of

Encrypted Web Browsing Traffic," in Proceedings of the IEEE Symposium on Security and Privacy, 2002.

[6] T. Kohno, A. Broido, and kc Claffy, "Remote Physical Device Fingerprinting," in Proceedings of the IEEE Symposium on Security and Privacy, 2005.

[7] S. Coull, C. Wright, F. Monrose, M. Collins, and M. Reiter, "Playing Devil's Advocate: Inferring Sensitive Information from Anonymized Network Traces," in Proceedings of the Network and Distributed System Security Symposium, February 2007.

[8] "LANDER: Los Angeles Network Data Exchange and Repository." `http://www.isi.edu/ant/lander/`.

[9] R. International, "PREDICT Project Web Page." `http://www.predict.org`.

[10] J. Mirkovic, "Privacy-Safe Network Trace Sharing via Secure Queries," in Proceedings of ACM CCS Workshop on Network Data Anonymization, October 2008.

[11] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: the second-generation onion router," in Proceedings of the 13th conference on USENIX Security Symposium - Volume 13, SSYM'04, (Berkeley, CA, USA), pp. 21–21, USENIX Association, 2004.

[12] "Tor Project: Anonymity Online." `https://www.torproject.org/`.

[13] "Request for comments: 1928." `https://www.ietf.org/rfc/rfc1928.txt`.

[14] S. Gebert, R. Pries, D. Schlosser, and K. Heck, "Internet access traffic measurement and analysis," in Proceedings of the 4th International Conference on Traffic Monitoring and Analysis, TMA'12, (Berlin, Heidelberg), pp. 29–42, Springer-Verlag, 2012.

[15] L. Sweeney, "k-anonymity: a model for protecting privacy," International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, vol. 10, no. 5, pp. 557–570, 2002.

[16] "pyinstaller." `http://www.pyinstaller.org/`.

[17] R. Pang, M. Allman, V. Paxson, and J. Lee, "The devil and packet trace anonymization," ACM SIGCOMM Computer Communications Review, vol. 36, no. 1, pp. 29—38, 2006.

[18] R. Pang and V. Paxson, "A High-level Programming Environment for Packet Trace Anonymization and Transformation," in Proceedings of ACM SIGCOMM, 2003.

[19] R. Lippmann, J. Haines, D. Fried, J. Korba, and K. Das, "The 1999 darpa off-line intrusion detection evaluation.," Computer Networks, vol. 34, no. 4, pp. 579–595, 2000.

[20] M. L. Laboratory, "DARPA Intrusion Detection Evaluation." `http://www.ll.mit.edu/IST/ideval/`.

[21] M. Mahoney and P. Chan, "An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection," in In Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection, pp. 220–237, Springer-Verlag, 2003.

[22] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," ACM Transactions on Information and System Security, vol. 3, no. 4, pp. 262–294, 2000.

[23] C. Thomas, V. Sharma, and N. Balakrishnan, "Usefulness of darpa dataset for intrusion detection system evaluation," in Proceedings of SPIE, vol. 6973, pp. 69730G–69730G–8, Spie, 2008.

[24] R. Chen, I. E. Akkus, and P. Francis, "Splitx: High-performance private analytics," SIGCOMM Comput. Commun. Rev., vol. 43, pp. 315–326, Aug. 2013.

[25] K. P. N. Puttaswamy, R. Bhagwan, and V. N. Padmanabhan, "Anonygator: Privacy and integrity preserving data aggregation," in Proceedings of the ACM/IFIP/USENIX 11th International Conference on Middleware, Middleware '10, (Berlin, Heidelberg), pp. 85–106, Springer-Verlag, 2010.

[26] A. Nandi, A. Aghasaryan, and I. Chhabra, "On the use of decentralization to enable privacy in web-scale recommendation services," in Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society, WPES '13, (New York, NY, USA), pp. 25–36, ACM, 2013.

[27] "The Internet Traffic Archive." `http://ita.ee.lbl.gov/`.

[28] "MAWI Working Group Traffic Archive." `http://tracer.csl.sony.co.jp/mawi/`.

[29] CAIDA, "Internet Measurement Data Catalog." `http://www.datcat.org/`.

[30] "Cooperative Association for Internet Data Analysis." `http://www.caida.org`.

[31] U. of Dartmouth, "CRAWDAD — a Community Resource for Archiving Wireless Data At Dartmouth." `http://crawdad.cs.dartmouth.edu/`.

[32] E. Kenneally and k. Claffy, "Dialing privacy and utility: A proposed data-sharing framework to advance internet research," Security Privacy, IEEE, vol. 8, pp. 31–39, july-aug. 2010.

[33] "Lobster web page." `http://www.ist-lobster.org/publications/deliverables/D1.1a.pdf`.

[34] G. Iannacone, "CoMo: An Open Infrastructure for Network Monitoring — Research Agenda." `http://como.intel-research.net/pubs/como.agenda.pdf`.

[35] G. Tech and U. of Napoli Ferderico II, "Project BISmark." `http://projectbismark.net/`.

[36] P. Eckersley, "How unique is your web browser?," in Proceedings of the The 10th Privacy Enhancing Technologies Symposium (PETS 2010), (Berlin, Germany), pp. 1–18, Springer-Verlag, July 2010.