Security Behavior Observatory: Infrastructure for Long-term Monitoring of Client Machines

Alain Forget, Saranga Komanduri, Alessandro Acquisti,
Nicolas Christin, Lorrie Faith Cranor, and Rahul Telang

July 14, 2014

CMU-CyLab-14-009

CyLab
Carnegie Mellon University
Pittsburgh, PA 15213

Security Behavior Observatory:
Infrastructure for Long-term Monitoring of Client Machines

Alain Forget[a], Saranga Komanduri[b],
Alessandro Acquisti[c], Nicolas Christin[d], Lorrie Faith Cranor[e], Rahul Telang[f]
Carnegie Mellon University
[a]aforget@cmu.edu, [b]sarangak@cs.cmu.edu,
[c]acquisti@andrew.cmu.edu, [d]nicolasc@cmu.edu, [e]lorrie@cmu.edu, [f]rtelang@andrew.cmu.edu

***Abstract*—Much of the data researchers usually collect about users' privacy and security behavior comes from short-term studies and focuses on specific, narrow activities. We present a design architecture for the Security Behavior Observatory (SBO), a client-server infrastructure designed to collect a wide array of data on user and computer behavior from a panel of hundreds of participants over several years. The SBO infrastructure had to be carefully designed to fulfill several requirements. First, the SBO must scale with the desired length, breadth, and depth of data collection. Second, we must take extraordinary care to ensure the security and privacy of the collected data, which will inevitably include intimate details about our participants' behavior. Third, the SBO must serve our research interests, which will inevitably change over the course of the study, as collected data is analyzed, interpreted, and suggest further lines of inquiry. We describe in detail the SBO infrastructure, its secure data collection methods, the benefits of our design and implementation, as well as the hurdles and tradeoffs to consider when designing such a data collection system.**

I. Introduction

Our understanding of the security and privacy challenges users face has grown substantially since some seminal usable security papers were first published [1], [2]. Much of the empirical data relating to topics such as authentication [3], [4], computer warnings [5], phishing [6], [7], identity theft [8], has been collected through either in-lab or online controlled experiments, or with surveys and interviews. Controlled lab and online studies allow researchers to isolate variables to observe and measure specific phenomena and effects. Survey and interview data have given us a better understanding of users' perceptions and perspectives, which are invaluable if we are to make security and privacy systems more usable. However, lab studies often lack ecological validity, since users may behave differently in the real world than in an artificial experimental setting [9]. Furthermore, self-reported data may not match users' actual behavior [10], [11].

Thus, the research community has begun focusing on more ecologically-valid data collection. Most published field studies to date have concentrated on specific sub-areas in the usable security and privacy field (e.g., text passwords [12], [13], ATM usage [11], malware infection [14], [15], mobile locking [16], social networks [17]). Most of these studies have short-term focus and monitor only a specific aspect of user or machine behavior. If we are to discover the ground truth of users' most pressing security and privacy challenges, it seems important to collect data on users' and their computers' overall naturalistic behavior in the wild over an extended period of time.

In this paper, we present and describe the Security Behavior Observatory (SBO) we designed to help researchers collect more ecologically-valid data of the widest possible scope over several years. The SBO is a client-server infrastructure for collecting data from a panel of several hundred household computers. Our software will allow us to deploy modular and independent sensors to monitor many security and privacy aspects of home computer use. Observing comprehensive and real-time decision-making of a large panel of users over an extended period of time in a real world setting, in itself, is invaluable. This information can provide a variety of practical and powerful insights into improving security and privacy policies and technologies. However, designing and building the SBO requires attention to factors less frequently considered in shorter-term, more focused studies. The infrastructure must be sufficiently scalable, reliable, and robust to collect the required size, breadth, and depth of data over the study's lengthy duration. In addition, we must carefully consider how best to maintain the security and privacy of participants' data, given the sheer amount and detail of behavioral data we will collect. We also require the flexibility to adjust the types of data we collect throughout the study, since research needs will invariably change as earlier data analysis leads to further lines of inquiry.

This paper is organized as follows. Section II describes how this project contributes to the science of security. Section III introduces the SBO and provides examples of the data we intend to begin collecting. Section IV elaborates on the SBO's architecture from two perspectives. First, we use a data flow model (Figure 1) to describe how data is collected from participants' client machines and sent to our server, and describe the specific benefits of our design decisions. Second, we use a deployment model (Figure 2) to describe our server configuration and how it securely and reliably handles the data encryption, transfer, and storage procedures. We briefly describe how participants enroll in our study in Section V. Section VI discusses some challenges, trade-offs, and limitations to consider when designing and deploying such an SBO system. Finally, we describe related work of similar data collection endeavors in Section VII and offer some concluding remarks in Section VIII.

II. The Science

Our understanding of computer and user behavior, with respect to security and privacy, has largely been based on studies of short duration and narrow focus. These studies have helped guide research over the past 20 years. However, a large-scale field study permits the measurement of users' security and privacy challenges and behaviors with much greater ecological validity than in the lab, where the experimental setting might not reflect users' actual behavior in their natural environment [**?**]. Furthermore, a long-term longitudinal study would provide data on the frequencies at which users encounter various security and privacy issues. These frequencies would represent risk probabilities, which are a key element of any risk assessment or risk management strategy. Thus, data from such a field study could be used to both inform and prioritize future research agendas.

To fill this need for more ecologically-valid data, we have built the Security Behavior Observatory (SBO): a framework for collecting data from a large panel of end-users whose online behavior will be monitored and analyzed over an extended period of time. This project is now possible thanks to widespread access to broadband Internet connections with reasonable upload speeds. The SBO offers an unprecedented window on real-time, real-life security and privacy behavior in the wild. Through the SBO, we aim to contribute to the evolution of a data-driven science of information security, with immediate applications in usability, economics, and secure system design. We hope this project will encourage discussions on collecting ecologically-valid data in current research practices, and serve as a template for future field studies.

III. Security Behavior Observatory

The Security Behavior Observatory (SBO) is a client-server architecture where participants' client computers are monitored over an extended period of time and upload collected user and computer behavior data to our servers. The initial launch of the SBO will monitor computers running Windows Vista, 7, and 8. We currently focus on these operating systems because their underlying architectures are almost identical (at least for the purposes of data collection), Windows has been the most popular operating system for the past 5 years [18], and desktop usage remains dominant over mobile computing [19]. However, the high-level infrastructure design and our own implementation (both described throughout this paper) can easily be applied to other operating systems (see Section IV-A7). Examples of the data we intend to monitor from hundreds of client machines over several years, with IRB approval and under strict security and privacy safeguards, include those described in the following subsections.

Our architecture is designed to provide data covering as much of the security and privacy space as possible. Some example research questions we intend to examine include:

- How up-to-date are operating systems?
- How long before a clean machine is infected, and how does infection actually occur in the wild?
- What are users' online social network privacy settings? Do they ever change, and why?
- What warning dialog messages do users encounter most often, and how do users respond?

As of this writing, we are performing final tests on most the implemented data collection sensors (see Sections III-A to III-E) while the others are under development (see Sections III-F to III-H). We intend to invite participants to complete questionnaires and interviews to elicit their perspectives on issues and events we observe throughout the study. We are beginning a pilot study on the main client-server SBO infrastructure (see Sections III-I to IV-A) and user study methodology (see Section V). We have purchased the server deployment configuration (see Section IV-B) and hope to begin data collection no later than this summer.

A. Filesystem

As currently designed, the SBO tracks changes to the filesystem, including the added, modified, or deleted file's size, last date modified, permissions, and other related information. [1] This data will help determine, for instance, if malware exists on the system and if so, how it affects machines' file systems, and whether or not users are likely to have noticed its presence.

B. Installed software and operating system updates

The SBO maintains a list of installed applications, their version numbers, and other related data, to determine what privacy or security software (e.g., anti-virus, firewall, adblockers, anonymizers) are installed, and whether they are up to date. The SBO also tracks which (and how soon after their release) operating system updates and patches have been installed. This allows us to measure the duration and severity of client machines' vulnerability to security threats.

C. Processes

The SBO monitors which processes (e.g., programs, applications) are running on clients' machines. It captures when all processes start and terminate, and can provide additional process status information at regular intervals. Primarily, this data will assist with the detection of malware. The SBO also collects general computer usage statistics that may help prioritize future security and privacy work, such as towards frequently-used applications.

D. Security-related events

The SBO also notes general security-related events, such as account-related events (e.g., logins, settings changes, password changes), registry modifications, wireless network authentications, firewall changes, and potential attacks detected by the operating system. This will provide valuable insights on multiple usable security topics, including the security measures users' employ on their computers, potentially dangerous program behavior, and the types and frequency of attacks that occur on home users' machines.

[1] However, we do not collect file or network packet contents since this may be too invasive and bandwidth intensive.

E. Network traffic

The SBO captures all network packet headers sent and received to clients' computers. [1] This data would allow us to detect various network traffic types that may be risky (e.g., peer-to-peer file transfers, dangerous websites) or suspicious (e.g., malware, intrusion attacks). We could thereby verify whether risky Internet behavior is correlated with a higher probability of an attack or infection.

F. Internet browsing behavior

We intend to further monitor users' web browsing behavior by collecting data from Microsoft Internet Explorer, Mozilla Firefox, and Google Chrome. We intend to capture search queries, online social network activity, browsers' and some online accounts' privacy and security settings, as well as other behavior of particular research interest (e.g., social networks, behavioral advertising). One example of possible analyses includes: what are users' privacy settings and behaviors on online social networks, do said settings adequately preserve users privacy, and if not, how could the website be better designed to empower users to more easily and accurately express their desired privacy settings. Another example of planned analysis consists of measuring how often users' actually make purchases derived from behavioral advertising links. This would reveal insights on the actual utility users gain from behavioral advertising, with respect to the privacy cost.

G. Configuration of software and online accounts

We also intend to track the security and privacy settings of users' software (see Section III-B) and online accounts (e.g., Facebook, Twitter). This would provide data regarding users' security and privacy practices. Should users change any such settings during the course of the study, it will be particularly interesting to understand users' motivation for initiating the change. If this could not be inferred with our data (i.e., if we did not detect any particular event preceding the setting modifications), we may send participants a survey or request an interview to inquire further.

H. Warnings

We intend to capture the content of and users' response to warning dialogs that request users make a security- or privacy-related decision. Past research has shown that users frequently do not understand these warnings, let alone know how to respond [5], [20]. This data would bring insights into the warnings users must cope with most frequently and what security and privacy decisions users make when prompted.

I. Security, Privacy, Usability, and Research Requirements

To capture such a wide array of data types over a long period of time, it is crucial we design and build an infrastructure that satisfies several requirements. First, we should minimize the impact of our data collection software on participants' computing and network performance. Thus, since the amount of data we can gather and transmit from clients is limited, we need the ability to be selective with and vary the types of data we collect over time. Second, as we collect and analyze data, we expect our research questions will evolve and require different types of data to be answered. For these reasons, our data collection architecture must be flexible enough to accommodate our changing needs. Third, unlike most experimental software which is typically used for only a short time for specific targeted purposes and environments, any problems caused by our client software could profoundly impact participants' computing experience, due to the breadth, depth, and duration of our data collection. Thus, our system requires a much higher degree of stability and reliability than typical experimental software. With these requirements in mind, we have designed and implemented the following architecture for the SBO.

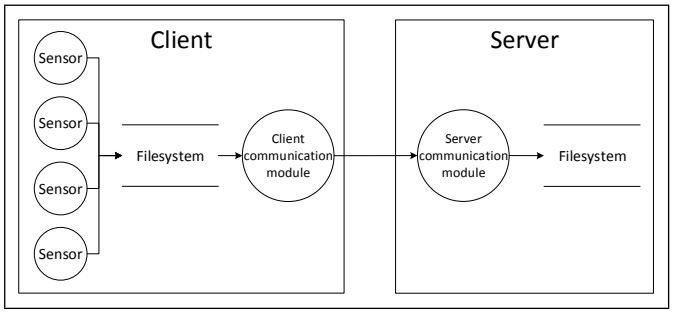

Fig. 1. Data flow between our SBO client and server software.

IV. Architecture

In this section we describe our design and implementation of the SBO architecture from two perspectives. We first illustrate how the data flows from initial collection on the client to storage on our server. Second, we discuss our deployment of servers and each of their roles. For both of these perspectives, we highlight the specific benefits of our design.

A. Data Collection and Flow

Figure 1 shows a data flow diagram of the client-server architecture. Each type of data is collected by a *sensor*, which outputs the data into a common directory. The *client communication module* periodically checks this directory for data files, and compresses, encrypts, and sends them over an SSL-encrypted channel to the *server communication module*. This architecture provides a number of beneficial design features.

1) Silent updates: We use Windows Installer [21] to package all the client software components into a single executable. Windows Installer provides functionality for cleanly installing and uninstalling the software, as well as upgrading. When the client communication module establishes a connection to the server, it first verifies that client software is up to date. If the server determines that it is not, the server provides a link to the current version's installation executable (hosted on our server) to the client. The client then disconnects from the server, downloads the current version of the client software, and checks the file's integrity with an MD5 hash. If the file is intact, the client shuts itself down after silently running

the installer executable in the background. Windows Installer then performs a "major upgrade" whereby the previous version is completely uninstalled before installing the new version. This clean-install approach avoids potential complex problems that can occur with minor upgrades and patches, which can result in an unstable software state. Should the update fail for some reason, Windows Installer will roll back to the previous software version, and the data collection can continue until the client attempts the update again. The entire update process is completely invisible to the user, and does not affect their normal computer usage in any way.

2) Independent sensors: Each type of data of interest (see Section III) is collected by a software *sensor* we have designed and implemented. Each sensor is independent of the rest of the data collection system. This sensor independence provides the following robustness and adaptability benefits. Firstly, if a sensor fails, the other sensors will continue to collect data, which the client communication module will continue to upload to the server. Secondly, if the client communication module fails or the server is unavailable, the client sensors will continue to collect and store data locally, and upload the data once the client communication module has finished restarting and/or the server becomes available. Thirdly, as the data interests for the study change over time, sensors can be silently (see Section IV-A1) and independently added, enabled, configured, disabled, or removed by the experimenters at any time without impacting any other aspect of the client system or our software. Finally, sensors can be implemented in whichever language is best for collecting the desired data. In Windows, this is most often a .NET language (e.g., C#, PowerShell), a command-line batch script, or Java.

3) Least privilege: To ensure clients' security and privacy, the principle of least privilege should be followed whenever possible. However, some data we seek to collect is likely to require administrator access to the client system. Fortunately, our architecture's sensors are independent, so higher privileges can be given only to sensors that require them.

4) Minimal footprint: Since the study's primary goal is to *observe* computer users' typical behavior, we must take care to avoid experimental effects that may influence this behavior. Thus, users should not notice a decrease in computer or network performance during the study. We achieve this in two ways. First, we take care to avoid intensive processing or blocking access to system resources as much as possible. Second, we throttle our client software's data upload speed to at most 192 kilobits per second (kbps), which is half of the slowest upload speed of the least expensive home Internet service plan available (excluding dial-up) in our initial area of participant recruitment (see Section VI-D). A data transfer rate of 192 kbps is equivalent to about 1.44 megabytes per minute, which is not much bandwidth for on-going data collection. This further enforces a minimal footprint by requiring the experimenters to be selective about what types and richness of data we collect. Although necessary, prioritizing what data to collect can be challenging (see Section VI).

5) Minimal user interaction: The use of passive observation to avoid experimental effects also implies we must minimize any user interaction. Our sensors and client communication module execute as Windows *services* [22], which implicitly provides this benefit. A Windows service is an executable program that runs in the background. Similar to Unix daemons, services (or any process or thread they spawn) cannot display any form of user interface (since Windows Vista). Thus, should a program running as a service attempt to display anything to the user, it will not be shown. This acts as a safeguard to ensure that we do not influence the user's normal computing tasks. However, this can be a challenge should the experimenters purposefully desire to interact with the user. This may be desirable should the experimenters wish to test participants' behavior to some stimuli. If future research questions require this, the application containing the stimuli would run as a standard program, not as a service, and would be designed so that any disruption to the user is minimized. However, the stimuli, and any effects it may have on all data being collected, should be carefully considered.

6) Multiple user accounts: Participants' computers may have multiple accounts. The computer's owner may have a separate account for guests, or each member of a household may have a separate account on a common machine. It is crucial that our data collection software run regardless of which user account may be logged in. Fortunately, Windows services can be set to always run when the system starts, independently of which user(s) logs in or logs out. Since our sensors and client communication module run as services, they are assured to run irrespective of which users login. Standard (non-service) applications can also be executed at startup, regardless of which user logs in, by adding a value to the registry [23].

7) Portability: Although we are currently targeting only Windows machines, we may desire the flexibility to collect data from other operating systems (OSes). To do so, we would almost certainly need to write new sensors, since the Windows underlying architecture is completely different from Unix-based operating systems. However, the client and server communication modules are written in Java, and thus should be easily-portable to any OS.

B. Deployment

There are several high-level requirements the SBO must meet. It is crucial that the data is securely and efficiently collected from participants. The data must must also be as securely and reliably stored as possible. Finally, researchers must be able to access and work with the data with as little inconvenience as possible. Figure 2 illustrates the deployment of our server architecture we believe best meets these requirements. We describe below each physical server's role, how data flows from the clients to the various server machines, and the security precautions that are in effect throughout.

1) Data collection server: The data collection server's role is solely to receive data from clients, and periodically send said

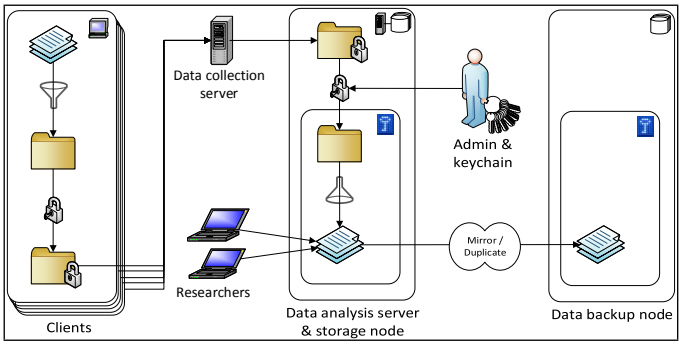

Fig. 2. Our SBO high-level hardware architecture and data flow.

data to the data analysis server when requested. The data flow from clients to the data collection server proceeds as follows:

1) Data is continuously generated on client machines (see Section IV-A).
2) At regular intervals, each client establishes an SSL connection to the data collection server.
3) The client and server mutually authenticate each other by encrypting random numbers with a shared symmetric authentication key [24].
4) When the server is ready to receive data, the client compresses the data, encrypts the compressed data with its symmetric encryption key (which is distinct from the authentication key and unknown to the data collection server), and sends it to the server.
5) The server stores the data locally, still encrypted with the client's encryption key.

2) Data analysis server: The purpose of the data analysis server is to periodically retrieve the encrypted data from the data collection server (and thereafter delete is from the data collection server), store all collected data in the data storage node(s), and provide access to researchers to perform work with the data. To ensure the data's security, it must remain solely on the data analysis server and be accessible only to project administrators and researchers. Thus, the data analysis server can be accessed only through a secure shell (SSH) tunnel originating from the specific IP addresses of the researchers' and administrators' work machines. To remotely access the data analysis server, researchers and administrators must first remotely connect to their work machine and, through said machine, establish an SSH tunnel into the data analysis server. Since the data must never exist anywhere other than our servers, all work with the data must be performed through this SSH tunnel.

As previously mentioned, the data analysis server periodically requests clients' encrypted data from the data collection server. This data transmission occurs over a mutually-authenticated SSL connection [24], and is scheduled to occur at a time of day when the data collection server is least likely to be busy receiving data from clients (e.g. 4:00 AM). The received data is still be encrypted with the corresponding clients' symmetric encryption key (see Section IV-B1). Clearly, the data cannot be analyzed while it is encrypted, but we also cannot risk storing it on the server unencrypted. Section VI-C discusses how we handle the decryption of the data for analysis.

3) Data node(s): Participants' encrypted data is ultimately stored in two places; in the data storage node(s) and data backup node(s). The backup node(s) are located in a physically-separate building from the storage nodes. These nodes are accessible only through the data analysis server, which represents the storage and backup nodes each as a network-attached storage (NAS) ZFS volume [25], [26]. Some key features of ZFS include snapshots (i.e., simple revision control), error detection, protections against data corruption, and storage pools, which allow the single logical ZFS volume to dynamically expand to include additional physical volumes. Thus, as our needs for additional storage grow and we add storage nodes, the additional storage space can simply be added to the existing logical ZFS volume, rather than being represented as a new volume (which would require additional researcher effort to manage and organize the data among multiple logical volumes).

An alternative filesystem could be the Hadoop Distributed File System (HDFS) [27], [28]. With similar benefits as ZFS, HDFS also allows data processing and analysis to be parallelized by distributing the data and computations among the nodes to more quickly process the data. However, HDFS cannot be treated as a traditional logical volume; it must be accessed through a special interface (i.e., API). Furthermore, programs must be written in a particular way to leverage the parallelism benefits of HDFS. Thus, Hadoop may require significant investment costs of time and effort. Furthermore, Hadoop would be beneficial when there are several data storage nodes which can perform computations in parallel. However, we currently need only a single storage node with an 8-core CPU to begin data collection, so the parallelism gains are not worth the time and effort investment. As the size of our panel and collected data grows to require several storage nodes, we will consider using Hadoop or another data storage and management technology instead of ZFS.

V. User Study Methodology

With the aforementioned infrastructure in place and having already obtained approval from our institutional review board for these procedures, we can solicit users to participate in our panel. Our primary method of finding participants is through a recruitment service for which people have asked to be notified about experiments. Potential participants will be asked a number of pre-screening questions. Participants must be over 18 and own a Windows Vista, 7, or 8 personal computer. We send interested persons an e-mail with a link to where they can complete the following initial enrollment tasks:

1) Reading and completing a consent form, which clearly informs users that we may monitor all activity on their computer and collect any data except for the contents of personal files, e-mails sent or received, content of documents on Google Docs, and bank card numbers.

2) Providing the names and e-mail addresses of others who also use the computer to be instrumented, so we can obtain their consent.
3) Completing an initial questionnaire
4) Download and install our data collection client software

Once these steps are complete, and all the other users of the computer have provided their consent, the participant is awarded a $30 Amazon.com gift card, since we can now collect data from the participant's machine. Participants thereafter receive a $10 gift card for every month our client software continues to upload data from their computer. This data transmission occurs silently in the background without requiring any action from participants. We also send periodic e-mails informing participants that either everything is working fine, which of the above enrollment tasks still need to be completed, or if we are not receiving data from their machine. If we do not receive data from users for 3 months, we may cease their participation.

VI. Discussion

There are a number of issues warranting careful consideration when collecting data from hundreds of participants' personal machines.

A. Participant IDs

It is necessary for our server to be able to identify which client belongs to which participant for several reasons. Primarily, every client machine must locally store its unique encryption and authentication keys to encrypt its data and securely communicate with our server (see Section IV-B1). We also need to verify users' continued participation (i.e. uploading data), so we can compensate them or remind them that they need to keep their computer on and connected to the Internet to continue participating. Additionally, we wish to be able to perform participant-specific data analyses to evaluate whether particular demographics are correlated with certain behaviors. We also wish to perform longitudinal analyses across specific machines' lifetimes (e.g. time before a malware infection).

The easiest way to identify client machines is to prompt the user for their assigned ID when they first install our client software. However, because our software runs as a Windows service (see Section IV-A5), it cannot display any user interface elements, and thus cannot interact with the participant. We solved this problem by creating an independent program that verifies that the stored participant ID and keys are valid, and if they are not, the program prompts the user. This program is run as a standard process, independently of any of our services, which allows it to interact with users if necessary. However, since the program does not run as a service, it does not execute within the same workspace or with the same privileges as the rest of the client software. Thus, we had to resolve various challenges regarding program-service communication, differing access control privileges, and synchronization.

B. Ethics & participant privacy

Although true for all user studies, it is critical that an institutional review board (IRB) approve the study's methodologies and procedures to ensure participants' are treated ethically and their data is kept confidential and secure. We spent considerable time iterating over our consent procedures with our IRB before their approval. However, many review boards do not have the expertise to understand the specific security and privacy challenges that may arise. Thus, the burden lies on the experimenters to consider carefully which data they are willing to collect and hold in trust, and to weigh the risk of a compromise with the value of such data to the advancement of the community's knowledge. Regarding de-identification, participants are assigned a random ID, which decouples their uploaded data from their provided personal information. We are also considering additional anonymization strategies and weighing their costs (e.g., loss of data richness, client-side computational loads) against possible threat models (e.g., client, network, server attacks).

C. Data Security

Given the potential sensitivity of the data our infrastructure collects and transmits from client machines across the Internet and stores on our servers, the data's security and confidentiality must be carefully considered and strictly enforced. In our implementation, we employ reliable end-to-end data encryption. Every client is assigned a unique encryption key. Client-side keys are stored in a permission-secured file on the client. To obtain the keyfile, an attacker would need access to the client with elevated privileges. The value of a participant's keys is unclear in this scenario, since this attacker could install malware to collect more sensitive information (e.g., passwords, bank account numbers) than we do.

Before transmitting the data, the client communication module compresses and encrypts the data with 128-bit AES [29] using Cipher Block Chaining mode [30] and PKCS5 Padding [31]. This encrypted data is sent to the server through an SSL connection and stored, still encrypted with the client's unique key. Once the encrypted data is received by the server, the client-side copy is deleted.

Although methods for computing on encrypted data exist (e.g., homomorphic encryption), our analyses across multiple sensors' data longitudinally across time are likely to be complex enough that they would not be practically feasible with such solutions. Instead, one researcher with access to all the clients' keys (stored in an isolated and secured MySQL database owned by a separate, dedicated, and tightly-secured user account) will decrypt and decompress each client's data into a TrueCrypt volume, to which all project researchers will have the key to analyze the data. Unencrypted data may temporarily exist in memory while and after working with it. However, the data must remain on the data storage nodes, which can be accessed only through a secure shell to the data analysis server from the specific IP addresses of the researchers' own campus machines. No other connections to this server are permitted. We feel that this is the best solution

for offering sufficient data security without overburdening researchers with complex, time-consuming procedures to access the data.

D. Client upload bandwidth

Given the wide breadth and depth of data we ideally wish to collect, the limit on how much data we can realistically collect is the clients' upload data rate. We must restrict the amount of data we upload from participants' machines to avoid a noticeable reduction of their Internet connection bandwidth.

To calculate this maximum upload rate, we first found the lowest data upload rate of Internet plans in our area, which is 384 kbps (kilobits per second). To avoid a noticeable impact on participants' network performance, we should use only a fraction of this total upload rate. By using only half, the actual data rate our software should be allowed to use is 192 kbps, or 24 KBps (kilobytes per second).

To ensure we do not surpass our desired bandwidth usable, we throttle clients' data upload speed by interleaving data transmission and sleep commands. For example, to achieve an upload rate of 24 KBps, the client process could alternate between uploading 6 KB and then sleeping for 250 ms until all the data is transferred. Sleeping between data transmissions should cause the OS to flush the uploaded data stream (i.e., actually send the data to our server rather than leave it in the client's network buffer in case our process adds more data to be sent) and free the network bandwidth and processing cycles for other applications until our application resumes. We hope to add adaptive throttling functionality to upload either more data when the network and computer are idle or less when the machine and Internet are in heavy use. However, this risks biasing the lower-priority types of data that would be collected only for clients with more computing capability and network bandwidth, which might be higher-income participants.

Given the massive amount of data this infrastructure can collect about client machine behavior (see Section III), it is important to employ techniques to minimize the physical size of the data transferred and stored. One such technique involves sensors that perform periodic snapshots, which should only log differences between the previously-recorded and current state, rather than always logging the complete current state. This is particularly important for snapshot sensors that gather large amounts of data for every snapshot. For example, when monitoring the filesystem, we may want to know all files' permissions, size, date first created and last modified, and potentially an MD5 hash. Such a complete list could easily be at least several hundred megabytes in size, which is an unreasonable amount of data to regularly transfer and store. However, only logging differences between the previous and current snapshot would likely be a realistically manageable size.

Another technique we use to reduce our data logs' footprint is the Binary JSON (BSON) data format [32]. The BSON data format is ideal for our infrastructure's purpose, since BSON is specifically designed to minimize spatial overhead in data transfers and storage, be easily traversable, and be efficient to encode and decode. We use BSON to log data that is either hierarchical in nature or may contain variable data elements (i.e., where there could be many null values if the data were logged in a flat table structure).

Even with data minimization techniques such as these, we anticipate having to make difficult decisions about which types of data to prioritize. However, while performing our preliminary data analysis, we may observe phenomena that we wish to further explore, but may be unable to because we had previously chosen not to enable the relevant sensors. To illustrate, suppose we initially choose to focus on malware infection. Thus, the minimum data we would need to collect appears to be network packet traffic, filesystem changes, and the executing processes. However, there are a number of scenarios where we would be missing data. For example, the network packet sensor would be unable to detect malware downloaded through SSL. Without the warning dialog sensor, we would not know if the user was ever warned from visiting a website, or prompted to download or install the malware. Without tracking security-related events, we may be unable to detect changes to the Windows firewall or other computer security settings. Admittedly, it may be possible to make some inferences from the sensors we did enable, but our understanding of the malware-infection events would certainly be incomplete. However, since we cannot possibly collect all the data, it is clear that there will be some limitations to the analysis we will be able to perform. Still, the choices of which data to collect in tandem will need to be made carefully, since poor decisions could pose unnecessary additional challenges when analyzing the data.

E. Server specifications & other cost considerations

There are a number of significant costs involved in conducting such a long-term data collection study. First and foremost, as discussed in Section IV-B, at least four physical server machines are required to begin the study; data collection, analysis, storage, and backup. Each of these machines have different specification requirements which should be carefully considered before committing to a purchase. The importance we place on each server's components are noted in Table I. Our reasoning for these priorities is as follows. The data collection server would benefit from several processor cores for receiving data from multiple clients at once, but these do not necessarily need to be the highest-possible clock speed (hence the *medium* rating). Our server software does not require much memory. This server's storage requirements are also relatively little, since it needs to retain only data collected over a few days, in case the data analysis server is temporarily delayed from removing the data (see Section IV-B2). The data analysis server itself also requires relatively little storage space for the operating system, data transfer, manipulation, and analysis applications and scripts. However, the data analysis server does require significant memory and processing power for its namesake purpose. The data storage nodes require at least a reasonably powerful processor and memory for data transfers to occur rapidly (and to quickly perform data processing tasks

TABLE I
IMPORTANCE OF EACH COMPONENT FOR EACH SERVER (SEE SECTION IV-B).

Server	Processor	Memory	Storage
Data Collection	Medium	Low	Low
Data Analysis	High	High	Low
Data Storage & Backup	Medium	Medium	High

if HDFS is in use). Of course, the storage nodes must have sufficient space to hold all the data to be collected. We use the following calculation to estimate our long-term storage needs. Assuming each participant uploads approximately 24 KBps (kilobytes per second) to our server (see Section VI-D), this equates to 377.4 gigabytes (GB) per year per participant. In our study's first year, we intend to have 100 users participating in our study. Thus, 100 participants each generating 377.4 GB per a year results in about 37.74 terabytes (TB) of data. We have obtained a minimum hardware configuration that satisfies the above requirements, and can be expanded for further data collection beyond one year, for around $35,000 (USD).

In addition to the server configuration costs, there are also on-going costs to be budgeted. Primarily, the participants require compensation. We are currently offering $30 for completing the necessary initial tasks to begin participating in the study (see Section V), and $10 for every month they continue to participate (e.g. we continue to regularly receive data from their machine). These costs add up quickly, since each participant costs $150 per year, so 100 participants cost $15,000 for a single year. Furthermore, if we cannot initially attract enough participants, we may need to consider increasing this stipend, which would further increase costs. Other on-going costs that should be considered include the technical administration and maintenance of the server hardware as well at least one dedicated project leader (and ideally a support team) to build and continuously refine the software and sensors, oversee the smooth execution of the study, and lead the data management (see Section VI-C) and analysis.

F. Study Limitations

Despite the wide scope of this infrastructure and study, there are some limitations which must be noted. Firstly, we are currently targeting only participants using Windows Vista, 7, or 8. Our focus on modern Microsoft operating systems (OS) means that we may not observe phenomena that occur on Unix-based OSes. Furthermore, mobile devices and tablets are growing in popularity [19]. Users' behavior and risk with respect to privacy and security with these devices may differ significantly than with traditional desktops or laptops. For future work, we could build sensors to collect data on Unix-based systems' usage, as well as mobile devices and tablets. Fortunately, our client communication module (see Section IV-A) can run on any system that supports Java (which includes most modern operating systems, see Section IV-A7).

In our user study, we ask users to install our software only on their one main Windows computer, because we are interested in observing the breadth of behaviors of multiple independent machines. However, people often have multiple devices through which they may have privacy and security challenges, including mobile devices and tablets. Thus, a complete in-depth examination of participants' behavior would require instrumenting all of a user's devices. This would be particularly challenging, given the multiple OS architectures participants may use. It is also unclear whether or not a participant's work machines should be instrumented. This would be required for a truly complete understanding of users' computing experience and behavior, but it would require participants' employers' consent, since data collection software on these machines may unintentionally capture the employers' intellectual property or other sensitive data. In any case, as our user study is currently designed, even though we capture a wider breadth of data than previous studies, we still risk missing some behaviors that occur on participants' non-instrumented devices. In future work, we hope to also collect data from mobile devices and tablets. We hope to reuse our client communication module to collect data from devices that support Java (see Section IV-A7).

As previously mentioned (see Section V), we offer participants $30 to complete the initial enrollment, and $10 per month of continued participation. This may bias our sample towards lower-income and privacy unaware or unconcerned participants. We will be able to confirm the former by asking participants to self-disclose their income in our enrollment questionnaire. However, it is unclear if any affordable level of compensation could attract higher-income participants. Additional compensation may also fail to attract privacy-concerned users, since users willing to be monitored are likely to do so for relatively small immediate short-term gains [33], [34].

VII. RELATED WORK

Lalonde Lévesque et al. [14] performed a 50-subject 4-month study of the effectiveness of an anti-virus software (AV) with respect users' computer behavior. Participants were given a Windows 7 laptop with Trend Micro's premium home anti-virus software and various monitoring software and scripts pre-installed. Every month, participants were required to meet with the experimenters to complete a survey about their computer usage and for the data to be collected from the machines. The AV detected 95 distinct threats on 38% of machines during the study, the vast majority of which were trojans, which is comparable with publicly-available statistics [14]. The authors' found 18 threats (e.g., 7 unwanted software, 9 adware, one malware, and another suspected as malware) that the AV failed to detect on 20% of machines. Participants with a greater computer expertise were more at risk of being exposed to threats than less computer-knowledgeable users. Furthermore, the authors reported that visiting sports and Internet infrastructure sites were more associated with a higher rate of infection, while visiting sites with pornographic or questionable content was less so. Although their methodology bares some resemblance to ours, there are several important differences between this and our study. Most obviously, our target sample size and study duration will both be several times

greater (i.e., hundreds of participants over several years). A more fundamental difference lies in our respective experimental models. Their study follows a "clinical trials" experimental model from medical research, whereby subjects are given a *treatment* (i.e., AV) and its effects are monitored over time. In contrast, our study's primary purpose is to passively observe our participants' and their machines' behavior by collecting a very wide array of security- and privacy-related data (see Section III) without any form of experimental intervention whatsoever.

Van Bruggen et al. [16] instrumented 149 student participants' Android smartphones with software that collected two types of data over two weeks; usage statistics (e.g., data usage, text messages, screen lock) and participant responses to weekly surveys on various topics. They found that 65% of their participants used a phone locking mechanism; 51% used the Android pattern lock and 14% chose a text password or PIN. They found no correlations for this choice with gender, previous phone type, text message frequency, data usage, or personality traits. Upon being surveyed about their password sharing behavior, 19% responded that they shared the password to their phone, while 63% shared passwords for other devices or services. The authors suggested that participants may place greater value the security of the mobile device over other devices or services. The authors later employed intervention messages based on incentives, morality, and deterrence to encourage users to either adopt a screen lock or upgrade to a more secure lock (e.g., from the pattern lock to a text password). The interventions did not appear result in many conversions. The authors concluded that the cost associated with targeting the users and implementing the interventions may not be worth the limited results. Our study does not currently target smartphones or attempt to modify users' normal computing behavior, we may consider testing attempts to assist, inform, and persuade users to take security precautions, should our data suggest that many users leave their computers dangerously vulnerable or otherwise behave insecurely. We also hope to expand our study in the future to include a broader range of devices, including smartphones and tablets.

Florêncio and Herley [12] collected Internet password data from over a half-million people over 85 days. This data was collected voluntarily from users of the Windows Live Toolbar. Their component hashed and stored passwords users' entered in web pages' password input fields, as well as the related URL, the passwords' bit strength, and other data. The authors also tracked incidents of password re-use as follows. Every time a character was typed into the web browser, their system hashed and compared each sequence of the last 7 to 16 typed characters to each of the stored password hashes that had been collected thus far. If a match was found and the current website's URL did not match the stored password hash's URL, then a password re-use event was logged. The authors reported many interesting findings of users' real-world password use, including the following highlights. Users had an average of 25 different online accounts, and typed 8 passwords on an average day. Users maintained an average of 6.5 distinct passwords, each across 3.9 separate websites. Users predominately chose lowercase-only passwords unless required otherwise. Finally, based on their study's results, the authors estimated that 0.4% of Internet users enter passwords on known phishing sites every year. Clearly, this study provided the research community with great insight into live user behavior, despite having only collected data for 3 months. However, unlike this study, we currently do not intend to collect data on participants' passwords (see Section VI-B), given the risks (despite our security precautions) of storing such data for a study spanning several years.

De Luca et al. [11] observed 360 people's interactions with automated teller machines (ATMs). A single experimenter personally monitored 60 people without their knowledge at each of 6 different banks' ATMs at varied times of day. The goal of the study was to better understand the context of ATM usage without capturing users' actual PINs. The data collected from each ATM interaction included the location, gender, time of day, interaction time, queue length, security measures taken by the user, and repeated PIN entry. The authors found that users were distracted in 11% of interactions, and that 65% of users made no effort to protect their PINs from observation attacks, either out of negligence, inability (e.g. carrying bags), or social context (e.g., did not want to imply mistrust in a nearby friend or family member). These and other results (including from interviews) led the authors to conclude that security should not rely on the user whenever possible, should be compatible with the social context, and PIN memorability is not a problem for most people, but it is severe when it occurs, since forgetting led to unsafe practices. The authors also shared lessons learned from the field observation study, including the utility of conducting pilot studies to test and refine the types and methods of data collection, abiding by strict codes of conduct to ensure ethical and consistent data collection, and importance of field studies in measuring users' actual behavior, which can differ from users' stated behavior in surveys and interviews.

VIII. Conclusion

Research to date has brought to light many usable security and privacy challenges computer users face, but there remain many unknowns, particularly with respect to home computer usages. Capturing data on these challenges in the wild as they occur naturally is essential if we are to conduct research and foster innovations with the greatest impact in improving the security and privacy of users and their machines. The Security Behavior Observatory (SBO) aims to collect said highly ecologically valid data on multiple security and privacy topics from hundreds of users' home computers over several years. This paper has specified the SBO client-server architecture, the benefits of our design decisions, and the challenges and trade-offs involved in building a system with the reliability, robustness, and flexibility required for a study of this lengthy duration and grand scope. We hope the data collected will yield insights on a wide variety of security and

privacy challenges, and guide future research efforts towards solving the challenges users actually face in the wild.

REFERENCES

[1] A. Adams and M. Sasse, "Users are not the enemy," *Communications of the ACM*, vol. 42, no. 12, 1999.

[2] A. Whitten and J. Tygar, "Why Johnny can't encrypt: A usability evaluation of PGP 5.0," in *USENIX Security Symposium*, 1999.

[3] R. Biddle, S. Chiasson, and P.C. van Oorschot, "Graphical passwords: Learning from the first twelve years," *ACM Computing Surveys*, vol. 44, no. 4, 2012.

[4] A. Forget, "A world with many authentication schemes," Ph.D. dissertation, School of Computer Science, Carleton University, 2012.

[5] C. Bravo-Lillo, L. Cranor, J. Downs, and S. Komanduri, "Bridging the gap in computer security warnings: A mental model approach," *Security & Privacy*, vol. 9, no. 2, 2011.

[6] J. Hong, "The state of phishing attacks," *Communications of the ACM*, vol. 55, no. 1, 2012.

[7] M. Jakobsson, "The human factor in phishing," *Privacy & Security of Consumer Information*, 2007.

[8] I. T. L. Review, "G.r. newman and m.m. mcnally," US Department of Justice, Tech. Rep. 210459, July 2005.

[9] M. Brewer, "Research design and issues of validity," *Handbook of research methods in social and personality psychology*, pp. 3–16, 2000.

[10] B. Berendt, O. Günther, and S. Spiekermann, "Privacy in e-commerce: Stated preferences vs. actual behavior," *Communications of the ACM*, vol. 48, no. 4, April 2005.

[11] A. De Luca, M. Langheinrich, and H. Hussmann, "Towards understanding ATM security – a field study of real world ATM use," in *Symposium on Usable Privacy and Security (SOUPS)*. ACM, 2010.

[12] D. Florêncio and C. Herley, "A large-scale study of WWW password habits," in *International World Wide Web Conference (WWW)*. ACM, May 2007.

[13] M. Mazurek, S. Komanduri, T. Vidas, L. Bauer, N. Christin, L. Cranor, P. Kelley, R. Shay, and B. Ur, "Measuring password guessability for an entire university," in *Conference on Computer and Communications Security (CCS)*. ACM, 2012.

[14] F. Lalonde Lévesque, J. Nsiempba, J. Fernandez, S. Chiasson, and A. Somayaji, "A clinical study of risk factors related to malware infections," in *Conference on Computer and Communications Security (CCS)*. ACM, 2013.

[15] N. Christin, S. Egelman, T. Vidas, and J. Grossklags, "It's all about the benjamins: An empirical study on incentivizing users to ignore security advice," in *International Conference on Financial Cryptography and Data Security (FC)*. Springer, 2011.

[16] D. Van Bruggen, S. Liu, M. Kajzer, A. Striegel, C. Crowell, and J. D'Arcy, "Modifying smartphone user locking behavior," in *Symposium on Usable Privacy and Security (SOUPS)*. ACM, 2013.

[17] G. Friedland, G. Maier, R. Sommer, and N. Weaver, "Sherlock holmes' evil twin: On the impact of global inference for online privacy," in *New Security Paradigms Workshop (NSPW)*. ACM, 2011.

[18] StatCounter.com, "Top 7 operating systems from july 2008 to nov 2013," October 2013, http://gs.statcounter.com/#os-ww-monthly-200807-201311.

[19] ——, "Mobile vs. desktop from july 2008 to oct 2013," accessed October 2013 2013, http://gs.statcounter.com/#mobile_vs_desktop-ww-monthly-200807-201311.

[20] S. Egelman, L. Cranor, and J. Hong, "You've been warned: an empirical study of the effectiveness of web browser phishing warnings," in *Conference on Human Factors in Computing Systems (CHI)*. ACM, 2008.

[21] Microsoft Corporation, "Windows Installer (Windows)," November 2013, http://msdn.microsoft.com/en-us/library/cc185688.aspx.

[22] ——, "Services (Windows)," October 2013, http://msdn.microsoft.com/en-us/library/windows/desktop/ms685141.aspx.

[23] ——, "INFO: Run, RunOnce, RunServices, RunServicesOnce and Startup," November 2013, http://support.microsoft.com/kb/179365.

[24] A. Menezes, P.C. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. CRC Press, 1996, ch. 10, p. 402, http://cacr.uwaterloo.ca/hac/.

[25] S. Watanabe, *Solaris 10 ZFS Essentials*, 1st ed. Prentice Hall, 2010.

[26] "OpenZFS," accessed November 2013, http://open-zfs.org.

[27] T. White, *Hadoop: The Definitive Guide*, 3rd ed. O'Reilly, 2012.

[28] Apache Software Foundation, "Welcome to apache hadoop," https://hadoop.apache.org/, accessed November 2013.

[29] Federal Information Processing Standards (FIPS), "Advanced encryption standard," National Institute of Standards and Technology (NIST), Tech. Rep. 197, November 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[30] ——, "Des modes of operations," National Institute of Standards and Technology (NIST), Tech. Rep. 81, December 1980, http://www.itl.nist.gov/fipspubs/fip81.htm.

[31] R. Laboratories, "Pkcs #5: Password-based cryptography standard," accessed November 2013, http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-5-password-based-cryptography-standard.htm.

[32] "Bson - binary json," accessed November 2013, http://bsonspec.org/.

[33] A. Acquisti, "Privacy in electronic commerce and the economics of immediate gratification," in *Conference on Electronic Commerce*. ACM, 2004.

[34] A. Shostack and P. Syverson, "What price privacy?" in *The Economics of Information Security*. Kluwer Academic Publishers, 2004.