Multimodal Data Fusion and Behavioral Analysis Tooling for Exploring Trust, Trust-propensity, and Phishing Victimization in Online Environments

Mickey Hefley
University of Nebraska at Omaha
mhefley@unomaha.edu

Gabrielle E. Wethor
University of Nebraska at Omaha
gwethor@unomaha.edu

Matthew L. Hale
University of Nebraska at Omaha
mlhale@unomaha.edu

Abstract

Online environments, including email and social media platforms, are continuously threatened by malicious content designed by attackers to install malware on unsuspecting users and/or phish them into revealing sensitive data about themselves. Often slipping past technical mitigations (e.g. spam filters), attacks target the human element and seek to elicit trust as a means of achieving their nefarious ends. Victimized end-users lack the discernment, visual acuity, training, and/or experience to correctly identify the nefarious antecedents of trust that should prompt suspicion. Existing literature has explored trust, trust-propensity, and victimization, but studies lack data capture richness, realism, and/or the ability to investigate active user interactions. This paper defines a data collection and fusion approach alongside new open-sourced behavioral analysis tooling that addresses all three factors to provide researchers with empirical, evidence-based, insights into active end-user trust behaviors. The approach is evaluated in terms of comparative analysis, run-time performance, and fused data accuracy.

1. Introduction

Malicious web content has bombarded user inboxes, social media feeds, and other online environments since the invention of the underlying supporting technologies. It is a problem that affects millions of web users daily and causes billions of dollars in yearly economic damage [1] to companies and individuals. Malicious content takes on a variety of forms including targeted (e.g. spear and whale) and untargeted phishing emails, social media posts, and websites that emulate the look and feel of legitimate sites. While spam filters and other technical measures prevent many phishing campaigns from reaching their targets, many millions of phishing emails and social media posts make it to their potential victims daily – putting the onus of phishing prevention into the hands of the end user. In 2012, approximately 156 million phishing emails were sent out world-wide *every day* [2], about 16 million made it through spam filters (despite a wealth of research and development), 8 million were opened and, in the end, 80,000 people became victims. In business environments, targeted phishing campaigns have been shown to be even worse, with success rates that netted up to 20% of users depending on the sector [3]. Clearly, phishing remains a human problem.

Security professionals and hackers have long known that humans are the weakest link in any cyber system. Adversaries tactically exploit user decision-making processes, preying on their *propensity to trust* based on certain learned contextual cues and environmental influences [4].

In previous work [5-7], we explored the human problem of phishing from the perspective of how and why structural elements in phishing attacks trick and victimize users. We designed a gamified experimentation platform, called *Cybertrust*, for presenting users with realistic web content (both malicious and innocuous) and investigating which types of content prompted suspicion and which types elicited trust. Using Cybertrust, and structural taxonomies of phishing content from the literature, particularly [8], we identified and published a set of structural trust factors, on nefarious and legitimate *antecedents of trust*, to classify malicious content based on which patterns of cognitive cues of trust and suspicion it contained. Using these patterns, we developed a *phishing victimization prediction model* [5-7]that pinpointed the effects of each pattern on victimization potential. Unlike heuristic models that emphasize economic advantage and user cognitive effort minimization as the primary driver for victimization [9], our model emphasized content structure as a victimization determinant. Validating the model ($p=0.01$) with an experimental study involving 80 subjects and over 5400 individual trust decisions, we showed that using the purely *a priori* structural patterns within content we could

URI: http://hdl.handle.net/10125/49995
ISBN: 978-0-9981331-1-9

successfully predict actual victimization for youthful populations of users (18-25 demographic).

While our previous work explored structural *antecedents of trust* as they related to the content, our platform suffered (like other studies and phishing research platforms [10-13]) from a lack of data collection and analysis robustness. In this paper, we define and exemplify a *data collection and fusion approach* that gathers user-centric behavioral data to support investigations of trust behavior that go beyond the content to examine *the person* interacting with it. With the goal of supporting future user studies into trust propensity, content trustworthiness, and training efficacy, the work described in the rest of the paper details the multi-perspective, multi-factor, data capture, fusing, and analysis tooling capabilities in what we have monikered "*Cybertrust 2.0*". Among other content-based data, the new Cybertrust now collects user interaction data including *eye tracking point of gaze, eye fixation points, 3D pupil modeling*, *mousing behavior*, *cognitive decision timing*, and *idle timing*. While techniques and tooling have long supported collecting such data, our approach is novel because it collects, fuses, and streams this data to an experimentation interface *in real-time*. The timeliness of this data fusion enables new analysis tooling, such as *real-time fixation-point/mousing heatmaps*, *point-of-gaze/mousing time-correlated graphs*, and *pupil-rate-of-change* visualizations that can be used to provide empirical, evidence-based, insights into active end-user trust behaviors.

The rest of the paper proceeds as follows. Section 2 identifies relevant background. Section 3 overviews the Cybertrust Platform. Section 4 describes the new data capture components created for Cybertrust 2.0. Section 5 defines our approach to fusing collected user-interaction data. Section 6 evaluates the approach and Section 7 concludes the paper.

2. Background

2.1. Trust, Victimization, and Phishing

When examining trust, it is important to identify and disambiguate the terminology. Our work uses terminology established in the foundational research of Mayer et al., in the so-called, *Integrative Model of Organizational Trust* [14] and empirical studies exploring this model, including those of Gill et al., [15]. In these works, *trust* is defined as "the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform a particular action important to the trustor, irrespective of the ability to monitor or control that other party" [14]. Here, the act of trusting is a unidirectional decision from *trustor* (risk taking party) to *trustee* (individual trusted by the trustor). This decision is preceded by *antecedents of trust* that include the behavioral profile of the trustor, the perceived characteristics of the trustee, and the nature of the interaction between the trustor and the trustee that precede the decision to trust. *Victimization* can occur if the trustee violates the trust of the trustor, particularly by misusing trust for gain at the expense of the trustor – for example in phishing campaigns.

Many behavioral factors are involved when trustors make trust decisions. Measures meant to predict an individual's likelihood to trust are often grouped into two categories: *experience* and *disposition* [12]. Disposition refers to an individual's inherent characteristics which are determined by numerous biological and environmental factors over the course of years. These characteristics include personal identity attributes such as *propensity to trust* [15], *honesty* [16], and *risk aversion* [17]. In contrast to dispositional characteristics, which broadly apply to different areas, experience is domain specific. For example, expertise with online environments is likely to affect an individual's trust in the cyber domain, but not in face-to-face communication.

In a study exploring these factors within the phishing domain [12], 299 subjects were given questionnaires measuring experiential factors including computer self-efficacy, web experience, and security knowledge as well as dispositional factors including trust propensity, risk aversion, and suspicion. Subjects completed coursework covering internet security and phishing and were reminded daily in class to never divulge their codes to anyone. At the end of the course, a phishing email was sent to each subject requesting their secret code. Approximately one third of the subjects in the study [12] failed to recognize the phishing attack and responded with their codes. Hence, *experiential factors have a much larger impact on phishing success than dispositional ones*.

Social presence is a concept defining the subjective and psychological experience of how information richness influences the persuasive effects of email. Harrison, et al. [18] constructed an experiment to examine the amount of social presence in email and its relation to phishing victimization. The information richness that they deploy shows how *sophistications* (i.e. content features that appear legitimate) can successfully dupe users by preventing them from noticing *degradations* (i.e. content features that are associated with malicious intent). Dhamija, et al., [19] investigate social presence and user awareness of website design. They found that attacks were more successful using various forms of visual deception that

rely on a lack of user knowledge of security indicators. Their work, which included 22 participants examining 20 legitimate and fraudulent websites, found that often people miss cues associated with trusted elements, such as certificates or padlocks in the browser, as well as cues associated with suspicious elements, such as an incorrect web address in the address bar. Findings in [20] suggest that these cues may be missed when a site includes sophisticated features that elicit trust, such as embedded brand logos in mass notifications from a recognized bank.

2.2. Eye Trackers Behavioral Analysis

Studies [21-24] have explored the connection between empirically observable eye movements and behavioral intention. Some [22], have specifically focused on *fixation point of gaze* as a correlate of behavior. *Heatmaps* and *fixation* paths [25] are visualization tools typically used to explore and analyze eye tracker data. These tools provide clear depictions of the areas of coverage and time-correlated traces of eye movements as they interact with content.

Eye trackers have been extensively used within the psychology [21, 23] and usability communities [24, 26] to help answer a number of behaviorally-oriented questions, such as what mental models govern the user's understanding of images and text [27] presented to them, or what screen space is used and/or wasted in the design of an interface [24].

While some eye tracking studies have explored trust behaviors in the context of economic decision making [23] and other similar trust scenarios, few have focused on phishing victimization. Among the few, Miyamoto et. al. captured eye movements as users examined phishing websites in a 2015 study titled *Eye Can Tell: On the Correlation between Eye Movement and Phishing Identification* [11]. Their study, post-hoc, identified inferential patterns of eye behavior that mapped to user intentions and outcomes (trusting or not trusting content). Inferential patterns were based on the number and duration of fixations. Their results demonstrate feasibility, but are limited to the sample size of participants and by the inability to analyze eye movement behaviors in-real time.

In another, earlier 2012 study, Kirlappos and Sasse [28] explored *trust seals*, i.e. seals of approval or validation such as Verisign® or DigiCert® using eye tracker data to determine if seals were effective to protect online shoppers. They found that users largely ignored trust seals and instead tended to rely on self-developed, but possibly unfounded ways of noting site trustworthiness, such as perceived site quality or references to other recognizable trusted entities.

2.3. Multitiered Data Fusion

Data fusion is a concept that largely originated in the 1990s with Hall and Llinas [29]. The core concept is simple: combine multiple data streams, usually sensor data, into a single stream to achieve some goals – which may be greater than the sum of the parts. In the context of security, several studies [30-32] have explored data fusion approaches to yield multi-perspective insights into security problems of interest. In dissertation work [32], Giacode examined data fusion in the context of visual analytics to determine the effectiveness of different sensor monitoring interface design parameters on conveyed security situational awareness. Another study[31], examined data fusion approaches for examining user clicking behaviors in the context of preventing click fraud – which is a type of fraud where a scripted tool clicks online pay-per-click online advertisements to generate revenue for the site owner (usually the same person).

Lastly, and most relevant to our work, Zhang et al, [30] examined data fusion approaches for building a phishing-site classifier. Their work used online data sets of textual and visual content in combination with a Bayesian model to create a classifier to detect phish. Data from text and visual analytics were fused together – which resulted in a better classifier than the sum of the individual classifier parts achieved. More specifically, the fusion algorithm selected the classifier (text, visual, or both) that held the largest probability of correctness, where *correctness* of each classifier was estimated using the Bayesian model.

3. The CyberTrust Platform

The Cybertrust platform is a multi-component, partially open-sourced, framework that supports multi-modal user interaction studies involving malicious online content. The platform, provides two main interfaces, tooling for creating experiment content (*admin view*) and the simulated online that study subjects interact with (*subject view*).

Admin view includes four major subcomponents: an *email creation tool*, *a social media post creation tool*, an *experiment workflow designer*, and *the behavioral analysis toolkit*. The first two components allow experiment designers to craft malicious and innocuous content items that study participants will later see. The third allows the designer to arrange those content items temporally into a *gamified workflow* that is meant to simulate the kinds of real-world tasks subjects do daily. The last component is a new addition made in this paper that provides a front-end interface

for researchers to explore captured subject data. The interface is discussed further in the next section.

On the subject side, study participants have access to several key components, including: the *task pane* which acts as a game driver to move task performance forward, an *inbox view* which emulates the look and feel of Google's Gmail® interface to display and allows users to interact with emails, and a *social media feed* that emulates the look and feel of Twitter®. Figure 1 shows a close up of an email in inbox view as a subject would view it in the platform. Pathways such as hover over and certificate inspection tools are available to users to simulate realism. Current work is underway to interpose a *global training interface* that spans these components to enable training feedback.

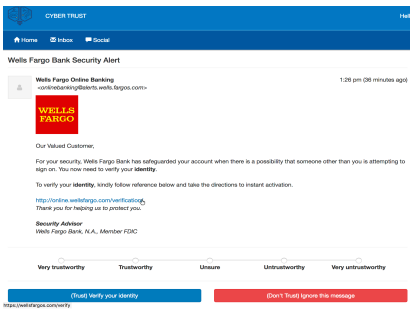

Figure 1. Malicious email in inbox view.

Cybertrust 2.0 has been architecturally redesigned to capture, stream, and aggregate subject behavioral data for use in phishing trust and victimization studies. The new design, as shown in Figure 2, is composed of several Docker containers and a *client app*. The first two containers reside on the *backend* as shown on the right side of Figure 2. The first container exposes an Apache webserver running a Django [33] RESTFul API in a python kernel. The API maintains endpoints to 1) serve client requests for experiment *tasks* to allow subjects to interact with the malicious and legitimate content of interest in the experiment and 2) endpoints that capture and persist user analytic data (*eye events*, *mouse events*, and *app events*). The second container encapsulates a SQL database that the Django API uses as a persistence layer for task and user behavioral data.

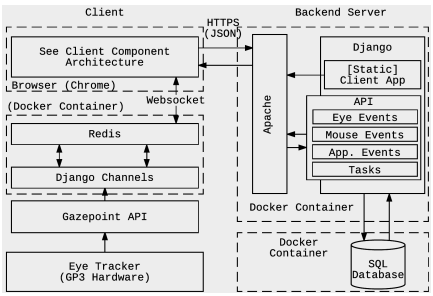

Figure 2. Cybertrust 2.0 containerized architecture

The client side, shown on the left side of Figure 2, includes a physically-connected Gazepoint GP3 device, another Docker container, and the client app in a Chrome browser. The GP3 streams eye movement events, discussed further in Section 4.3, to its open API. The last container collects, converts, and standardizes data from the GP3 API before streaming it to the client app using websockets. To handle the rapid, asynchronous, event processing required to send and consume websocket messages, we use *redis* [34] and *Django Channels*, an asynchronous plugin for Django. The last portion of the client, is the Cybertrust 2.0 app as shown architecturally in Figure 3.

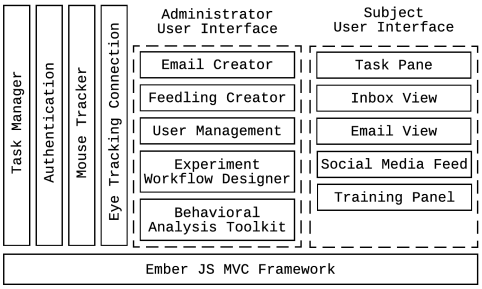

Figure 3. Client component architecture

Implemented in Ember.js [35] and running in a Chrome browser, it provides the administrator and subject-facing interface components discussed above. The app also includes several new components that manage data collection and review (i.e. *Mouse Tracker*, *Eye Tracking Connector*, and *Behavioral Analysis Toolkit*). The first two are information capture elements that collect and aggregate data during subject sessions. The third is a post-hoc review tool that administrators can use to review captured subject data. We discuss these further in the next section.

4. Capturing Empirical Measures of Trust

Information capture capabilities are critical in any platform that seeks to understand and reason about the fundamental behavioral underpinnings that govern end-user trust decisions. The new Cybertrust 2.0 platform captures end user interaction data including mouse and keyboard events, application-level events related to trust decision making, as well as eye tracking events. These data provide multiple evidence-oriented perspectives that can be used to identify how user decisions materialize when engaging with malicious, untrustworthy, content. Each Cybertrust 2.0 information capture mechanism is defined below, in terms of the specific subfields captured and the technical tooling involved in the capture.

4.1. Mouse and Keyboard Events

Upon logon, the platform begins to track all mouse and keyboard events. The volume of events captured is associate to the intensity of the user's interaction with the application. As defined below, a typical capture varies from 15hz to 60hz. Our platform uses and extends an open source JavaScript tool [36] to capture the mouse and keyboard events below.

Mouse Movement: On a user mouse movement event, capture {*x, y, w, h, t*}, where *x* and *y* identify the pixel coordinates of the mouse, in two-dimensional screen space, $x \leq w$ and $y \leq h$), *w* and *h* define the maximum screen size (in pixels), and *t* is the time when the mouse event occurred. Our capture mechanism relies on a custom implementation of the *onMouseDown* event, that uses the W3C standard *high resolution time level 2* specification [37] to provide much more precise timings. The standard provides worst-case resolvable timing skew of 5µs. Other mouse and keyboard events also use high res. timing.

Mouse Selection: On a user mouse up event, capture {*ht, t*} where *ht* is highlighted text and *t* is the high-resolution timestamp when the event occurred. If no text is highlighted this event is ignored.

Mouse Clicks: On user mouse down capture {*x, y, w, h, t*}. Similarly, to mouse movement, *x* and *y* define pixel positions in screen coordinate space, *w* and *h* define the max screen width and height, and *t* is the timestamp when the click event occurred.

Key Press: On a user key down event, capture {*k, t*}, where *k* is the Unicode character identifying the key pressed (as translated from its browser-specific *keycode*) and *t* is the time the event occurred.

4.2. Application-level events

When a user session begins, the application begins logging trust decisions and timing events as users engage with tasks. Since application level events are high level and associated with task performance, they are captured much less frequently than mouse and eye tracker data.

Session Timing: Upon subject login, capture {*sts*, *ste*}, where *sts* and *ste* are high-resolution timestamps identifying the start and end times of the user's session.

Time tracking (task): When a subject views task content (e.g. views an email) capture {*tts, tte, tid*}, where *tts* and *tte* are the start and end times that identify how long the user views content associated with the *task*, with id *tid*.

Likert scale: After a user views task content (e.g. views an email) but before they make a binary trust decision they must rate the trustworthiness of the content, by selecting *very trustworthy*, *trustworthy*, *unsure*, *untrustworthy*, or *very untrustworthy*.

Binary trust decision: After a user rates the trustworthiness of the content, they must either trust or distrust the content by clicking a button with a contextually relevant message. Once a binary decision is made capture {*tid, lrt, tr*}, where *tid* is the task id, *lrt* is trustworthiness captured by the Likert scale and *tr* is a Boolean for *trust* (true) or *do not trust* (false).

Idle time: When a user goes into an idle state, capture {*it, et*}. Where *it* and *et* are high resolution timestamps identifying the start and end times of user idling. A function executes periodically to determine if another event has occured, if not a new *it* value is logged. When the next event occurs the time is logged as *et*.

4.3. Eye Tracker Events

Upon logon, the Cybertrust platform initiates an eye tracker session with the Gazepoint GP3 HD [38] and begins logging *eye movement* events, as defined below, at 150hz. The GP3 exposes data using an open XML-based API. The hardware driver for GP3 also normalized the data as the subject moved around. In our work, we created a new, open-source, websocked-based interfacing server that converts GP3 events to JSON and streams them to the Cybertrust 2.0 client.

Eye Movement events are logged following the schema: {*tid*, *cnt, et, t-tick, msg, cx, cy, fpogid, fpogd, fpogs, fpogv, fpogx, fpogy, bpogx, bpogy, bpogx, lpogx, lpogy, lpogv, rpogx, rpogy, rpogv leyex, leyey, leyez, lpupild, lpupilv, reyex, reyey, reyez, rpupild, rpupilv, lpcx, lpcy, lpd, lps, lpv, rpcx, rpcy, rpd, rps, rpv}* where *tid* is the application-level task that the user is currently interacting with (if any); *cnt* is internal record counter (record id) used by the Gazepoint GP3; *et* is the system *Datetime* stamp corresponding to the current system time; *t-tick* is a signed 64-bit integer which identifies the number of CPU time ticks involved in the event and is used by windows' Query Performance Counter (QPC) for high-resolution timing based on CPU time stamp counters (TSC); and *msg* is an optional field that can be used to 'tag' an event with textual data (such as debug information). Other values include *cx* and *cy* which identify mouse *x*, *y* positions (in screen coordinate space) detected by the GP3. These are useful for correlating to application detected events, discussed in Sections 4.1 and 4.2.

The fields *fpogid, fpogd, fpogs, fpogv, fpogx,* and *fpogy,* are parameters related to the *fixation point of gaze* which identifies the *id* of the fixation point, the *starting timestamp* (*s*) when it begins, the *duration* (*d*) it lasts, the *x* and *y* pixel positions in screen coordinate space where the user is fixating, and a Boolean *validity* value (*v*) indicating whether the fixation is valid or not – based on pupil analysis algorithms on the GP3 hardware. The fields *bpogx, bpogy, bpogx, lpogx, lpogy, lpogv, rpogx, rpogy,* and *rpogv* are *point of gaze* variables for the *best* (*b*), *right* (*r*), and *left* (*l*) pupils. Essentially, the GP3 computes individual points of gaze for the right and left eyes and then algorithmically combines them for the *best* (average) point of gaze. In each of the *b*, *r*, and *l* fields: **pogx* and **pogy* are the (x,y) pixel positions of the point of gaze and **pogv* is a Boolean value similar to *fpogv* that identifies if the point of gaze is valid or not. The validity value is 0 (false) if the subject blinks or the pupil is obscured and 1 (true) otherwise.

The fields *leyex, leyey, leyez, lpupild, lpupilv, reyex, reyey, reyez, rpupild,* and *rpupilv* are related to the 3D position of the eyes. The fields **eyex*, **eyey*, and **eyez* identify the (x,y,z) coordinate positions of the left and right eyes, relative to the GP3 camera focal points. The other fields **pupild* and **pupilv* represent the *diameter* (*d*) of the eye and whether the measurement is *valid* (*v*) for the left and right eyes.

Finally, the remaining fields, i.e. *lpcx, lpcy, lpd, lps, lpv, rpcx, rpcy, rpd, rps,* and *rpv,* represent image data regarding the positioning of the left and right pupils. The **pcx* and **pcy* values stand for pupil center x and y coordinates that identify the center of the left and right pupils in the image snapped by the camera, as a fraction of the camera image size (note there are 150 images captured per second). The **pd* value identifies the *pupil diameter* in the image. The **ps* value represents the scale factor of the pupil. When users are at the calibrated distance, the scale factor is 1. If they move back from their calibrated position, the factor is greater than 1. Inversely the value is less than 1 if they move closer to the device. The scale factor is useful for ensuring that users stay positioned correctly and for correcting measurements if they do not. Lastly, the **pv* measurement is a Boolean indicating the validatity of the pupil data.

Overall, our websocket-based streaming data client integration allows the Cybertrust 2.0 platform to consume eyetracker events in near-real time. The specific timing information regarding the precision of this data in the client is discussed later in Section 6.3. This near-real time streaming enables several training and analysis directions unexplored in the phishing victimization prevention literature, e.g. just-in-time training that highlights areas indicative of suspicion.

5. Fusion Techniques for Trust-relevant Interaction Data

Taken individually by capture mechanism, each data stream empirically identifies facets of behavioral patterns that users reveal as they make trust decisions. When fitted together, the pieces begin to fuse into an overall picture of behavioral tendencies. In this section, we describe our data fusion approach and a few real-time analytics tools enable research directions that go beyond post-hoc analysis. While the visualizations underpinning the tools, i.e. heatmaps and event graphs, are not novel, the timeliness and applicability to behavioral studies investigating phishing victimization set the tools apart. These novel aspects are just demonstrative of the type of outcomes that the data capture and fusion approach can provide.

Central to our approach is a universal event timestamping strategy. The same internal system clock is used by all capture mechanisms, so that we can accurately fuse data streams into a single view, without the worry of distortions due to clock skew. The specific precision of data capture mechanisms is discussed later in Section 6.3. For now, know that each event capture has a time resolvability that the platform uses along with the timestamp to arrange all captured events into a workflow that allows a researcher to view events in real-time during user studies.

Fused data is time-correlated and interlaced into a single stream. The stream itself is a three-layered model as shown in Figure 4, where each layer is timeline of events collected from a particular data capture stream (eye tracking, mouse tracker, and application data). Following this model, a cross sectional slice in time spanning the three 2D planer layers defines events related to a trust decision. Using a multilayer approach, such as this, provides researchers with the analytical flexibility to view a layer independently or in conjunction with others.

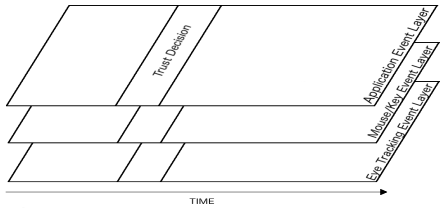

Figure 4. Fused three-layered data streams in time.

In addition to the layering, the fusion happens in real time, according to the timing of the data capture streams. This enables novel application use cases, such as real-time heatmap and event graph creation for real-time training feedback, or real-time experimentation interjection by researchers. This can enable new behavioral study designs that interpose real-time

human or machine-in-the-loop feedback within or during task performance to explore the effects on mental model adjustment or trust antecedent considerations. Although interesting, such exploration is out of scope and left to future studies.

To simply demonstrate the feasibility of real-time data analytics, we created two real-time analytics tools. The first, produces *event graphs* across the three layers in the data fusion model. These event graphs are nothing more than real-time stream timelines that researchers can view in real time as events are captured during subject task performance.

The second tool is a *real-time heatmap generator* that researchers can use to visualize heatmappable user data. Heatmappable data includes eye tracking events and mousing events. Both contain x, y coordinate pairs that can be plotted in real time as users generate events of certain types. To implement this capability, we make a series of RESTful API requests to the backend container and then open a websocket from the backend to the researcher (admin) client that, respectively, retrieves previous data, and then allows them to consume all new incoming, trust relevant, interaction data generated during the active user session. Once all the necessary data is loaded and as it continues to stream in, JavaScript is used to fuse the events together in the multi-layer model. After fusion occurs, the output, is piped to visual components (either the event graph or heatmap) to be displayed. The researcher can choose which data to plot by picking from the layers.

These analytical tools represent a step forward in empirical measurement of cognitive reasoning in phishing victimization studies. For example, analyzing eye tracking data, one can observe if a subject was focusing on the content defects or its sophistications, where sophistications are given by the content classification from [8], then act upon those observations by providing the subject with targeted training to help them recognize those problems in the future. Using mouse tracking data, one can observe if the user highlighted suspicious content, hovered over a link, or idled for a period of time. With his data, it might be possible to gain better insight on how they came to a decision and use the insight to mitigate similar victimization-causing decisions in the future.

6. Evaluation

To evaluate the data collection and fusion approach we compared it against other platforms and studies from the literature, assessed its run-time performance, and measured the precision of captured data. The methodology and results for each evaluation type are defined in the next sections.

6.1. Comparative Analysis

In the first form of evaluation, we compared the Cybertrust platform to five other studies in the literature. Comparisons were made according to three factors: the *contextual realism* of experimentation environment, *trust antecedents* investigated, and the *types of interaction data gatherable in each*. Table 1 overviews this comparison, decomposing each factor into sub-factors (rows) according to the study explored (columns). Checkmarks indicate the factor is supported or present in the study and X's indicate that it is not. Realism is rated from Low (L), to Medium (M), to high (H). All values are color coded, where red is bad and dark blue is best.

Table 1. Feature and data collection capabilities comparison across studies.

	Cybertrust	Anti-Phishing Phil [8]	Whalen [5]	Darwish [34]	Miyamoto [6]	Halevi [39]
Realism / Trust Antecedents						
Level of Realism	M	L	M	M	L	H
In the Wild	X	X	X	X	X	√
Email / Inbox	√	X	√	√	X	√
Web sites	X	√	X	√	√	X
Social Media	√	X	X	√	√	X
Embedded Images	√	X	√	X	X	X
URL Inspection (hover)	√	√	√	X	X	√
Mouse interaction	√	X	√	√	X	√
Certification Inspection	√	X	√	√	√	√
Attachments	√	X	X	√	X	√
Popups	√	X	√	X	X	X
HTML Support	√	X	√	√	X	√
Interaction Data Captured						
Mouseing History	√	X	X	X	X	X
Key Press History	√	X	X	X	X	X
Idle Timing Log	√	X	X	X	X	X
Content Inspection Timing	√	√	X	√	X	X
Trustworthiness Likert	√	X	X	X	X	X
Trust / Did Not Trust Decision	√	√	√	X	X	√
Eye Movements	√	X	√	√	√	X
Fixation Point of Gaze	√	X	√	√	√	X
Right/Left Point of Gaze	√	X	√	√	√	X
Single Best Point of Gaze	√	X	√	√	√	X
Pupil / Eye 3D positioning	√	X	√	√	√	X
Pupil Dilation	√	X	X	X	X	X

The first, Anti-Phishing Phil, is an online game designed to teach young people anti-phishing habits through a fish and worm gamified metaphor [8]. Players try to 'eat' good worms and avoid bad worms to get points in the game. The second, higher fidelity comparison point, was a study by Whalen and Inkpen investigating users' browser security habits. Their work captured eye tracking data and screen activity such as mouse movements and scrolling as users engaged with content in a browser [5]. Another study, by Darwish and Emad, explored user behaviors exhibited while viewing online logon pages [34]. Perhaps the closest study to our work, was one

conducted by Miyamoto et al., that explored the connection between victimization the structure of legitimate and phishy websites. In their work screenshots of both types of websites were presented to users and eye trackers were used to log eye movements [6]. The final study we examined in our comparative analysis, conducted by Halevi, et al. [39] is prototypical of industry standard 'in the wild' testing. In the wild studies of this type actively phish users by sending internal emails that contain links that, when clicked, enroll employees in remedial training.

6.2. Data Capture / Fusion Performance

Data capture and fusion performance was examined in the second evaluation. Two testing environments were used as benchmarks: *client* and *backend*. The client environment used in this study was a Windows 10 machine with the following system specifications: Intel® Core ™ i7-6700k CPU @ 4.0 GHz, 4001Mhz, 4 core (8 logical cores), 16GB RAM (DDR-2400Mhz), Windows 10 v10.0.14393, Chrome v58.0.3029.110 (64bit), Sandisk x300 SSD (6.0GB/s). The 'backend' environment was a Docker container allocated 3 CPU Cores (2.9GHz) and 4GB RAM (1867 MHz DDR3).

Client test data, discussed below, was gathered using the Google Chrome developer toolkit[1]. These tools profile the in-browser performance of the application. To collect containerized backend test data, we used Google's open source tool cAdvisor (or Container Advisor)[2] in conjunction with Docker.

Two evaluative criteria were used. Both criteria touch upon two user stories in Cybertrust: As an ***end user***, I want to ***view and rate online content***, so that *I can learn to identify and avoid phish*; and As an ***experiment designer***, I want ***end-user data to be captured and logged***, so I can *better understand the behavioral intricacies of phishing victimization*.

Criteria 1: The end-user test subject should not observe any performance hits during data collection and logging and should be able to complete tasks (e.g. view and rate online content) unimpeded.

Criteria 2: All captured data points collected during a subject's session should be sent to, parsed, and stored by the backend data API server without loss.

To test the performance along these two criteria, we simulated typical end-user in a typical data capture session. The notion of 'typical' was based upon previous studies [5-7], where we noted that content examination tasks rarely exceeded 30 seconds in duration. Using this as a baseline, we simulated end-user mousing data using a test script that fed random mouse movement events to the system at the hardware level. The browser environment detected these movements as user interactions. During this same time, a tester sat in front of the Gazepoint GP3 eye tracker and randomly looked around the screen to generate eye tracker events. We repeated this test 30 times and averaged the results.

Across the 30 tests, the average client reaction time to a user interaction event, during full data collection load, was 68ms. **This met Criteria 1.** Full load collection consisted of (on average) 150 events per second of eye tracker data and 95 events per second of mousing data (this is determined by the randomness of the mousing script, but largely maps to realistic human behavior [26]). This amounts to 4500 eye events and 2850 mouse events per 30 second task. Each eye event data record was, on average, 1020 Bytes (in JSON API format) and each mouse event record was, on average, 452B. This amounted to 4.59MB of eye data and 1.29MB of mouse event data. Figure 5 shows an example client capture session (1 of the 30 runs) of client test results as they appear in the profiling tool.

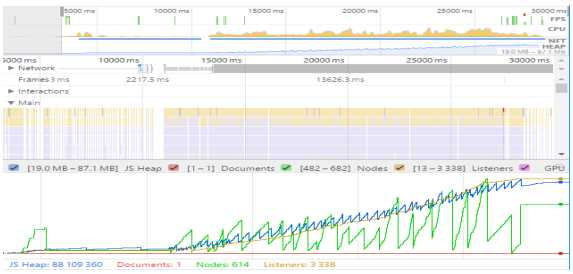

Figure 5. 1 of 30 Client session captures showing (top to bottom) CPU and FPS, network request frequency, user interactions, JS call stack (main), and JS Heap, listeners, and DOM nodes

Despite the low size of data records, during task performance, the Javascript heap size grew, on average, by 271 MB. This was largely due to the high number (3340 on average) of listener callback functions that tended to stack up over time as part of the ajax-based API invocations. Since each event generated a network request there were 4500+2850 backend API calls (of type *application/vnd.api+json*), or 245 network request per second. This turned out to be an issue because the client queued up network requests and which began to stack up as the session time went on. This is shown in Figure 5 as a mostly solid grey bar in the *network* row.

While this queue delay did not impact the client responsiveness (i.e. Criteria 1) or the server performance it caused issues for data integrity (i.e.

[1] https://developer.chrome.com/devtools

[2] https://github.com/google/cadvisor

Criteria 2). In fact, it took more than 4 minutes, on average, after the 30 second task performance session completed, for the client to catch up and issue all of the ajax requests placed in its queue from the task session. The issue is that if the user closed their browser, all queued requests would be lost.

On the backend server side, the performance of the API was admirable. Figure 6 shows a capture of container performance in one of the 30 tests as monitored and reported by cAdvisor. Across the 30 tests, CPU utilization maxed out at 33% (at full load). Memory usage did not exceed 150MB and was, on average less than 100MB, much less than the 4GB allocated to the container. Average total request handling time (per request) was 30ms. The one issue encountered on the backend was in the database Docker container. Here, the high number of requests per unit time resulted in 8 database locks (SQLLite). Based on this issue and the queue delay problem, the platform **did not meet Criteria 2** and more work was needed to deal with these inefficiencies.

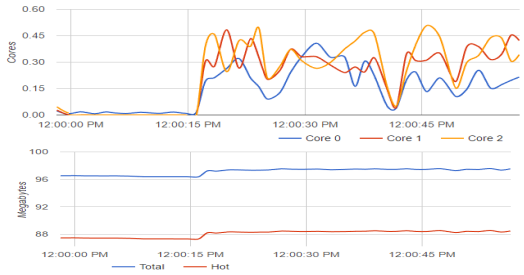

Figure 6. Backend container CPU (top) and memory (bottom) performance during a test run.

To mitigate these problems, we made two changes: one to the client and one to the backend. On the client-side we instituted a batch-commit strategy that accumulated 150 eye events before committing and 100 mouse events (roughly 1 second's worth). This immediately reduced the number of network requests down to 1.95/s (instead of 245/s). Overall, this increased the average backend request handling time to 350ms, due to the parsing required to handle the batch data, but reduced the network queue on the client side. This reduced the number of average listeners per 30s user session to 1200 (mostly associated with the application itself, instead of ajax callbacks), and the JS Heap size to a maximum 60MB. It also drastically decreased the main call stack – since the application did not need to queue network requests. On the server, our data collection strategy resulted in ~250 INSERT queries per user per second. Hence, we switched from SQLLite to PostgreSQL, a much more robust SQL database backend capable of handling hundreds of thousands of queries per second. With PostgreSQL, we can support around 275 simultaneous user data capture sessions on the test backend configuration and can scale up vertically with higher memory allotments. With the combined changes, **Criteria 2 was met**.

6.3. Post-fusion Data Precision

Using the client test environment described in Section 6.2, this section evaluates the timing precision of fused data collected and aggregated by Cybertrust 2.0. Timing precision analysis is important when capturing unique data streams running in various disparate environments (such as in-browser data captures vs eye tracker hardware/api/container). For this reason, precision is evaluated for each information capture type described previously in Section 4.

The first class of data, i.e. mouse and keyboard events, all use the same capture timing mechanisms. This includes mouse movement, mouse selection, mouse clicks, and key press. As discussed briefly in Section 4 each of these event handlers uses Javascript's high resolution time module, i.e. performance.now(), latency, to resolve event timings to the nearest 5µs. Using the profiling tools in Chrome, we found that our platform achieves an average timing accuracy of 800µs for mouse and keyboard events. This timing includes the latency between initial event firing and data logging. The timing is also much more accurate than using the *Date* library [37] which is subject to system clock skew of up to 300ms.

Application level event data have various time precisions. The first, *idle time* is significantly less accurate than any other event capture mechanism in Cybertrust 2.0, but is still highly accurate with respect to user behavioral timings. It has an average accuracy of 100ms. Unlike other types of event captures in Cybertrust, idle time is a measure of when something *isn't happening*, so it must wait to observe other event handlers to ensure none of them are firing. This means it is subject to the asycnronicity and length of the event processing queue. If no handlers execute within a 100ms timing interval, a handler in Cybertrust executes to log an idle event. As soon as another user event handler executes, the idle handler executes again and logs the idle time length to the backend.

The *time tracking* data type is, on average, accurate to 805µs. This is because the start time, *tts*, is not subject to event handler latency, and therefore has a timing precision of 5µs. The parameter **tte,* marking the end of a task, is subject to the same event handler latency associated with the mouse and keyboard events – since the event is not processed until its handler is called, which occurs on average after 800µs.

The last time-sensitive application event data type is *session timing*. The starting timestamp, *sts,* is

accurate to 5µs, while the ending timestamp, *ste*, is accurate to 5 minutes (if the user does not actively 'logout' and *ste* uses the server-side session timeout) at worst, or 800µs (if the user actively logs out). Users will be asked to logout when their session is complete to alleviate this issue.

The last type of data, eye events, is by far the most precisely accurate. Events generated by the GP3 have an on-average time resolution of 333 nanoseconds. This order of magnitude difference compared to the mouse and key data capture precision is based on the windows Query Performance Counter data collected by the GP3 in its *t-tick* field. This data allows the GP3 to precisely identify the moment of capture based on the CPU cycle data collected in windows.

7. Conclusion

This work defines a three-layered data fusion model along with information capture mechanisms capable of gathering, aggregating, and streaming user interaction data in real-time during end-user phishing content assessment. Our approach provides researchers with better investigative tools for exploring and analyzing phishing victimization. Using a three element evaluation, we show that the approach is performant, produces precise, fusable data, and exceeds the information capture capabilities of other studies and platforms in the literature.

8. References

[1] "The Cost of Phishing: Understanding the True Cost Dynamics Behind Phishing Attacks," *Cyveillance news*, 2009.

[2] "315,000 new malicious files every day," *Kaspersky press*, 2013.

[3] Verizon, "2016 Data Breach Investigations Report," 2016.

[4] K. Kreijns, P. A. Kirschner, and W. Jochems, "Identifying the pitfalls for social interaction in computer-supported collaborative learning environments: a review of the research," *Computers in human behavior,* vol. 19, no. 3, pp. 335-353, 2003.

[5] M. L. Hale, C. Walter, J. Lin, and R. F. Gamble, "Apriori Prediction of Phishing Victimization Based on Structural Content Factors," *International Journal of Services Computing,* 2016.

[6] M. L. Hale, R. Gamble, J. Hale, M. Haney, J. Lin, and C. Walter, "Measuring the Potential for Victimization in Malicious Content," in *22nd IEEE International Conference on Web Services*, 2015.

[7] M. L. Hale, R. F. Gamble, and P. M. Gamble, "CyberPhishing: A Game-Based Platform for Phising Awareness Testing," in *48th Hawaii International Conference on System Sciences*, 2015.

[8] J. Staggs, R. Beyer, M. Mol, M. Fisher, B. Brummel, and J. Hale, "A perceptual taxonomy of contextual cues for cyber trust," in *Colloquium for Info. System Sec. Education,* 2014.

[9] X. R. Luo, W. Zhang, S. Burd, and A. Seazzu, "Investigating phishing victimization with the Heuristic–Systematic Model: A theoretical framework and an exploration," *Computers & Security,* vol. 38, pp. 28-38, 2013.

[10] T. Whalen and K. M. Inkpen, "Gathering evidence: use of visual security cues in web browsers," in *Proceedings of Graphics Interface*, 2005.

[11] D. Miyamoto, G. Blanc, and Y. Kadobayashi, "Eye Can Tell: On the Correlation Between Eye Movement and Phishing Identification," in *Intl. Conf. on Neural Info. Processing*, 2015.

[12] R. T. Wright and K. Marett, "The influence of experiential and dispositional factors in phishing: An empirical investigation of the deceived," *J. of Mgmt. Info. Sys.,* vol. 27(1), pp. 273-303, 2010.

[13] S. Sheng, Mannien, B., Kumaraguru, P., Acquisti, A., Cranor, L.F., Hong, J., and Nunge, E., "Anti-Phishing Phil: The Design and Evaluation of a Gamble that Teaches People Not to Fall for Phish," in *Symposium on Usable Privacy and Security*, 2007.

[14] R. C. Mayer, J. H. Davis, and F. D. Schoorman, "An integrative model of organizational trust," *Academy of management review,* vol. 20, no. 3, pp. 709-734, 1995.

[15] H. Gill, K. Boies, J. E. Finegan, and J. McNally, "Antecedents of trust: Establishing a boundary condition for the relation between propensity to trust and intention to trust," *Journal of business and psychology,* vol. 19, no. 3, pp. 287-302, 2005.

[16] P. Dasgupta, "Trust as a commodity," *Trust: Making and breaking cooperative relations,* vol. 4, pp. 49-72, 2000.

[17] T. H. Chiles and J. F. McMackin, "Integrating variable risk preferences, trust, and transaction cost economics," *Academy of management review,* vol. 21, no. 1, pp. 73-99, 1996.

[18] B. Harrison, A. Vishwanath, Y. J. NG, and R. Rao, "Examining the Impact of Presence on Individual Phishing Victimization," in *48th Hawaii International Conference on System Sciences*, 2015.

[19] R. Dhamija, Tygar, J.D., and Hearst, M., "Why Phishing Works," in *ACM Conference on Human Factors in Computing*, 2006.

[20] M. Blythe, Petrie, H., and Clark, J.A., "F for Fake: Four Studies on How We Fall for Phish," in *ACM Conference on Human Factors in Computing Systems*, 2011.

[21] U. Park, R. Mallipeddi, and M. Lee, "Human implicit intent discrimination using EEG and eye movement," in *International Conference on Neural Information Processing*, 2014.

[22] Y.-M. Jang, R. Mallipeddi, S. Lee, H. Kwak, and M. Lee, "Human intention recognition based on eyeball movement pattern and pupil size variation," *Neurocomputing,* vol. 128, pp. 421-432, 2014.

[23] B. Colombo, C. Rodella, S. Riva, and A. Antonietti, "The effects of lies on economic decision making. An eye-tracking study," *Research in Psychology and Behavioral Sciences,* vol. 1, no. 3, pp. 38-47, 2013.

[24] R. Jacob and K. S. Karn, "Eye tracking in human-computer interaction and usability research: Ready to deliver the promises," *Mind,* vol. 2, no. 3, p. 4, 2003.

[25] D. D. Salvucci and J. H. Goldberg, "Identifying fixations and saccades in eye-tracking protocols," in *Proceedings of the 2000 symposium on Eye tracking research & applications*, 2000.

[26] M. C. Chen, J. R. Anderson, and M. H. Sohn, "What can a mouse cursor tell us more?: correlation of eye/mouse movements on web browsing," in *Conf. on Human factors in computing systems*, 2001.

[27] P. D. Allopenna, J. S. Magnuson, and M. K. Tanenhaus, "Tracking the time course of spoken word recognition using eye movements: Evidence for continuous mapping models," *Journal of memory and language,* vol. 38, no. 4, pp. 419-439, 1998.

[28] I. Kirlappos, M. A. Sasse, and N. Harvey, "Why trust seals don't work: A study of user perceptions and behavior," in *International Conference on Trust and Trustworthy Computing*, 2012.

[29] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," in *IEEE Intl. Symposium on Circuits and systems,* 1997.

[30] H. Zhang, G. Liu, T. W. Chow, and W. Liu, "Textual and visual content-based anti-phishing: a Bayesian approach," *IEEE Trans. on Neural Networks,* vol. 22, no. 10, pp. 1532-1546, 2011.

[31] M. Kantardzic, C. Walgampaya, B. Wenerstrom, O. Lozitskiy, S. Higgins, and D. King, "Improving click fraud detection by real time data fusion," in *Intl. Symp. on Signal Processing and IT,* 2008.

[32] N. A. Giacobe, "Measuring the effectiveness of visual analytics and data fusion techniques on situation awareness in cyber-security," The Pennsylvania State University, 2013.

[33] *Django*. Available: http://www.djangoproject.com/

[34] J. L. Carlson, *Redis in Action*. Manning Publications Co., 2013.

[35] J. Cravens and T. Q. Brady, *Building Web Apps with Ember. js*, O'Reilly Media, Inc., 2014.

[36] github.com/elwayman02/ember-user-activity, 2017.

[37] W3C, "High Resolution Time Level 2 Standard," 2016.

[38] Gazepoint, "Gazepoint GP3 HD," 2017.

[39] T. Halevi, N. Memon, and O. Nov, "Spear-phishing in the wild: A real-world study of personality, phishing self-efficacy and vulnerability to spear-phishing attacks," 2015.