Cross-Layer Personalization as a First-Class Citizen for Situation Awareness and Computer Infrastructure Security

Aokun Chen Pratik Brahma Dapeng Oliver Wu Natalie Ebner Brandon Matthews †
Jedidiah Crandall ‡ Xuetao Wei * Michalis Faloutsos *∗ Daniela Oliveira
University of Florida MIT Lincoln Laboratory† University of New Mexico‡
University of Cincinnati * University of California Riverside *∗
{daniela,wu@ece.ufl.edu}, {prprbr,chenaokun1990,natalie.ebner@ufl.edu}, brandon.matthews@ll.mit.edu
crandall@cs.unm.edu, weix2@ucmail.uc.edu, michalis@cs.ucr.edu

ABSTRACT

We propose a new security paradigm that makes cross-layer personalization a premier component in the design of security solutions for computer infrastructure and situational awareness. This paradigm is based on the observation that computer systems have a personalized usage profile that depends on the user and his activities. Further, it spans the various layers of abstraction that make up a computer system, as if the user embedded his own DNA into the computer system. To realize such a paradigm, we discuss the design of a comprehensive and cross-layer profiling approach, which can be adopted to boost the effectiveness of various security solutions, *e.g.*, malware detection, insider attacker prevention and continuous authentication. The current state-of-the-art in computer infrastructure defense solutions focuses on one layer of operation with deployments coming in a "one size fits all" format, without taking into account the unique way people use their computers. The key novelty of our proposal is the cross-layer personalization, where we derive the distinguishable behaviors from the intelligence of three layers of abstraction. First, we combine intelligence from: a) the user layer, (*e.g.*, mouse click patterns); b) the operating system layer; c) the network layer. Second, we develop cross-layer personalized profiles for system usage. We will limit our scope to companies and organizations, where computers are used in a more routine and one-on-one style, before we expand our research to personally owned computers. Our preliminary results show that just the time accesses in user web logs are already sufficient to distinguish users from each other,with users of the same demographics showing similarities in their profiles. Our goal is to challenge today's paradigm for anomaly detection that seems to follow a monoculture and treat each layer in isolation. We also discuss deployment, performance overhead, and privacy issues raised by our paradigm.

NSPW '16, September 26-29, 2016, Granby, CO, USA

DOI: http://dx.doi.org/10.1145/3011883.3011888

CCS Concepts

•**Security and privacy → Intrusion detection systems;**

Keywords

Intrusion detection system, Cross-layer personalization

1. INTRODUCTION

Protecting end-user devices and computer system infrastructure is an ongoing problem, and today's solutions have not evolved significantly over time. Specifically, the industry still mostly relies on antivirus-based solutions, which are almost exclusively using signature-based detection of threats [1–3]. Although showing good accuracy for known malware, these solutions cannot detect zero-day malware and encounter difficulties in identification of polymorphic and metamorphic attacks, with a practical detection rate of 25%-50% [4]. Behavioral-based solutions have also been adopted [5–12], focusing on identifying behavioral properties of the device, such as peculiar sequences of system calls, and use of this information to distinguish patterns that characterize malware. However, behavioral-based detectors are not always effective, presenting high false-positive rates [12, 13], because of the increasing complexity and diversity of software.

To make matters worse, malware is evolving, and current security solutions for computer infrastructure and situational awareness are not adequate to cope with the increasing level of sophistication of attacks. For example, organizations are now targets of advanced persistent threat (APTs) attacks, highly planned and orchestrated infiltrations combining different types of exploits and malware [14–16], which can be caused by insiders and outsiders. APT malware is challenging for current solutions because it blends in with approved corporate software and traffic, and because it acts slowly, based on triggers.

This material is based upon work supported by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract No. FA8721-05-C-0002 and/or FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Assistant Secretary of Defense for Research and Engineering.

Today's security solutions have two main shortcomings. First, they usually come in a "one-size-fits-all" format, as they are designed for "any" computer system, and make decisions about computer system events while ignoring the unique way different people use their computers. Second, these solutions usually operate at a specific layer of abstraction (*e.g.*, the application, the network or the OS syscall layer), and thus can miss correlated and insightful observation from other layers.

Nowadays, it is common for a computer system (*e.g.*, desktop, laptop, tablet, mobile phone) to have a single user and never be shared. Also computer systems' users are diverse and have rather different user profiles when using a computer. For example, some people just use a browser for their computing needs, while others use computers as part of their work and run specific applications at specific times, which might include personal and task-oriented software. For example, consider Bob, 72, living in a retirement community in Florida. His desktop usage pattern is rather different than that of Alice, 25, working for a startup in San Francisco. Bob uses a browser to read the news and e-mails and to check for events in his community. He also uses Skype to talk to his son in Illinois. Alice has MS Office installed on her laptop and regularly uses MS Word to write weekly reports for her manager. She also uses Chrome browser and regularly accesses sites hosted not only in the United States, but also in China. She works from home Mondays and Wednesdays, and on these days, VPN and VNC client are running for connection with the company server. While Bob never accesses his computer past 8 p.m., Alice is usually active on her computer up to midnight.

The patterns of usage for Bob and Alice suggests that computer systems behave as if they possess a personalized "microbiota." In biology, the microbiota is the large set of microbes that share our body with our human cells and outnumber human cells by a ratio of 10:1. The large majority of them are benign and influence, for better or worse, critical functions for our physiology, such as digestion, allergic reactions, and propensity to diseases, such as cancer or Alzheimer's. Each human's microbiota is unique, providing a signature for the individual, a signature which has been proposed to be applied even in criminal forensics [17]. The microbiota's constitution depends on the human's diet, lifestyle and geographic location. In spite of that, microbiota composition form clusters depending on people's similarity such as based on share living contexts, shared lifestyle, living in the same geographic area, or having similar healthy conditions.

Going back to Bob's and Alice's pattern of computer system usage and making an analogy to the human microbiota, we can compare their computers to the human body and the events happening in the system to the microbiota. We hypothesize that the idea of uniqueness and clustering also apply. While the patterns of usage for Bob and Alice form a signature that identifies them, their patterns of usage also form clusters when we compare them with similar computer users. For example, we hypothesize that Bob's computer profile is similar to other older adults, especially if they live in the same area–for example, a retirement community. Also, Alice's pattern of work shares similarities with colleagues working on the same project, as they probably use the same software, access the same files, and work similar hours.

In this paper, we propose a new security paradigm that makes **cross-layer personalization a first-class citizen of security solutions for defense of computer infrastructure and situational awareness. The phrase first-class citizen does not contain any form of discrimination, but best expresses our confidence in the effectiveness of this new paradigm, which we believe will be a premier component in the future security system design.**

In this paper, we discuss the architecture and the initial Windows OS implementation of this new paradigm, the machine-learning challenges for profile building with preliminary results, and our plans for evaluation of the system with a user study placed in the real world. Our Windows profiling system combines information from three layers of abstraction: 1) the user layer(*e.g.*, mouse clicks and frequency of keyboard activity); 2) the operating system layer, through process and file events; and 3) the network layer, through events and metadata associated with network events. The goal is to create a computer usage profile of a particular machine based on a particular user that operates it.

In this framework, a kernel module continuously collects the events that constitute the computer system profile. These events are used to build a multi-dimensional time series for each computer user. Depending on the goal of the security solution (*e.g.*, malware detection, outlier detection, continuous authentication, profiling based demographics, *etc.*), a specific machine-learning model is built using the data from the multi-dimensional time series. There are many machine-learning challenges for this problem. First the amount of data produced is very large and high-dimensional, and some information will be incomplete. Additionally, users might change the profile dynamically because of a change of habit, time zone, or projects. Further, an adversary, which can be the computer user himself, might also attempt to confuse the model and evade profiling by intelligently adding highly correlated noisy data.

Our initial results show that such system requires from 50 to 100 KB/day for generic process-related information and 3.5 MB/day for network-related information. The performance overhead caused by the extractor was negligible for a chess benchmark. We also discuss preliminary results obtained in the context of previously collected data in an independent user study, in which just the connection time of participants' web logs was used to distinguish user profile. Finally, we discuss the design of a field study to collect ecologically valid data to test our hypothesis of the uniqueness of computer system profiles and the effectiveness of using these individual profiles to protect organizations' computer infrastructure.

While there has been some related work in personalization in the areas of secure account login [18], user characterization based on web searches [19], and recommendation systems [20]. These previous approaches were designed only for the single-layer analysis and the focus was not on holistic computer infrastructure defense. We argue that cross-layer personalization should be a premier component in the design of any security solution for computer infrastructure. Furthermore, we argue that the personalization should be holistic, involving intelligence from all layers of abstraction (**cross-layer**). Our intention is to generate discussion on the full potential of such an approach, but also to debate its challenges and deployment concerns, which could include

privacy issues and the dynamic nature of usage profiles.

This paper is organized as follows. Section 2 describes our threat model. In section 3, we present the design of our paradigm. Section 4 describes our vision for the architecture of a system implementing the paradigm. Section 5 describes how we characterized the computer system usage profile and how we are extracting its data for the Windows OS. Section 6 describes the machine-learning methods we are planning to use and the challenges to be faced: a large, multidimensional, sparse, and dynamic data set. Section 7 describes preliminary work and outlines the design of a user study to collect data to validate our paradigm for the problem of malware detection. Section 8 discusses related work in personalization and malware detection. Section 9 summarizes the important topics discussed at the workshop. Section 10 concludes the paper.

2. THREAT MODEL

The target of our paradigm is the protection of organizations against attacks from outsiders and insiders. Outside attacks are represented by all types of standalone malware that can enter the organization perimeter via drive-by downloads, malicious links and attachments opened by employees, etc. For this type of attack, the cross-layer profiling can triage events for standard malware detectors by filtering events that are normal for a particular employee and by exposing true outlier events that should be carefully inspected, as they can be the result of malware activity.

The paradigm can also protect an organization against insider attacks. Once the profile of a group of employees working on the same project or mission is created, the runtime analyzer can expose employee behavior that is incompatible with the expected behavior of the group.

We envision the core components of our paradigm implemented at the OS-level because the OS has a privileged view of events occurring at the application, OS, and network layers. Consequently, the paradigm relies on a trustworthy OS.

Our vision for the implementation of the paradigm does not involve the recording of keys typed by the user, file contents, or any personal identifiable information. For example, as later described in Section 5, we propose the recording of the applications used by the user, their active time intervals, network activities and filesystem metadata. Our vision does not advocate the recording of file contents, keys types by the users, e-mail content, *etc.*

Furthermore, we assume that solutions based on our paradigm will be employed in organizations, where employees already have limited expectations of privacy. It is very common in modern organizations to monitor the traffic and e-mail, and restrict the devices and applications used by employees. Of note, we are not advocating that organizations should or should not perform fine-grained monitoring of their employees. Such discussion goes beyond the scope of our work. Our main point is that considering that such monitoring is already a reality. The collection of non-personal identifiable cross-layer data would not impact the current confidentiality expectations of employees in organizations.

The diversity that cross-layer personalized profiling offers makes it much harder for an adversary to succeed. The adversary would need to mimic benign events in a personalized fashion, which greatly increases the effort and the resources required for a successful attack.

3. THE PARADIGM

Our proposed paradigm makes cross-layer personalization a premier component in the design of security solutions for the defense of computer infrastructure and situation awareness.

This paradigm is based on the observation that computer systems have personalized usage profiles that depend on the users and their associated activities and the profiles span the various layers of abstraction that make up a computer system. Our paradigm combines: (a) cross-layer and (b) personalization solutions for computer infrastructure and situational awareness.

First, we argue that we need to monitor a computer at multiple layers, including OS, network, and application layers, whose definitions we refine later. Second, we need user-specific personalization to better detect atypical behaviors. We target user devices (or user accounts for multi-user machines), which are typically used by one or a few people in a fairly predictable manner. We envision that cross-layer personalization should be the next generation of security solutions for organizations, especially in the context of malware & insider attack detection and continuous authentication.

Why are cross-layer personalized security solutions better than standard ones? Because events that are typical or normal for one user might be anomalous for another, and a standard security solution will err when confronted such a variability in usage. It will either fail to recognize a malicious event or it will generate a false positive. Also, the abnormality might be the result of a correlation of events across more than one layer of abstraction, something that a standard solution focusing on one layer is not able to capture.

For example, connections to Chinese IP addresses at 11:30 p.m. are anomalous for Bob, but are normal for Alice. In contrast, heavy I/O activity and file accesses at 7 a.m. are anomalous for Alice, but normal for Bob, who wakes up early and helps with his homeowners' community affairs. Consequently, an anomaly detector running at Bob's and Alice's computer should learn their usage patterns and use this knowledge to distinguish typical and atypical events for Bob and Alice respectively, where atypical events might be an indication of a security violation.

As another example, consider a working group in an organization that accesses a set of common files, performs similar activities on these files, and uses a certain set of primary task software. If one of the group members starts deviating from the group profile, such as by accessing files from a directory the group usually does not access, or by connecting to servers the group usually does not connect to, this could signal the presence of a malicious insider credential theft. Thus, cross-layer personalization allows for the implementation of solutions performing continuous authentication, where the user profile adds another layer of authentication that is more difficult for an attacker to evade, because it is dynamic, unique in isolation, and similar within groups.

4. ARCHITECTURAL DESIGN

This section presents our design for the cross-layer profiling framework. Our architecture (Figure 1) contains an *Event Extractor* that continuously records system events related to user interaction with processes (*e.g.*, timestamps of mouse clicks and keys typed), and related to process,

file system and network activities. The profiler does not record keys, file contents or personally identifiable information. The extractor is located at the OS level because of the kernel's visibility of the application layer, network layer, and even low-level system events.

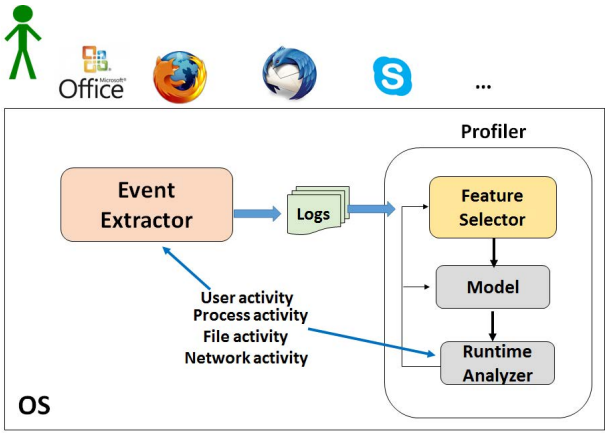

Figure 1: The architecture of the cross-layer profiling framework. The Event Extractor collects system events related to user activity, processes, file system and network, and records these events in Log files. The Log files are consumed by the Profiler, which generates a Model tailored to a particular security problem. The Runtime Analyzer observes the same events as the Event Extractor at runtime, classifies them according to the Model, and generates different types of outcomes depending on the type of security solution it applies to: anomaly logs, alarms, reports, intervention, *etc.*

The Event Extractor records events according to a data model and generates log files. Then, the *Profiler*, which is also located at the OS level, consumes these logs and generates a target model depending on the particular security problem at hand, for instance, malware detection, outlier behavior detection, user profiling, *etc.* The Profiler has three components. The first is the *Feature selector*, which uses machine-learning feature-selection algorithms to analyze raw data and–in combination with the characteristics of the security problem and, optionally, domain knowledge–eliminates features or creates new ones based on the collected data. For example, depending on the goals of the security solution to be generated, the Feature selector might disregard process-memory consumption as feature, and create a new feature, such as mouse click frequency, based on the time stamps recorded for mouse clicks. The second component is the *Model*. This component is trained by an unsupervised machine-learning algorithm based on all the features and pre-defined rules.

The third component is the *Runtime Analyzer*, which observes the same events recorded by the extractor at runtime and classifies them based on the model. Depending on the security problem at hand, the Runtime Analyzer produces an outcome (e.g. an alarm, an entry in a report file or log file, *etc*). The Runtime Analyzer also generates feedback for the Feature selector and the Model. For example, in insider attack detection or continuous authentication, an outlier event, depending on how much it deviates from the model, might indicate a change of routine or a security violation. Given the dynamic nature of the computer usage, the Profiler can incorporate acceptable changes into the model.

5. PROFILE EXTRACTION AND CHARACTERIZATION

The first step toward building the cross-layer profiling framework is to determine which data to collect across all layers of abstraction. Ideally, the extractor should collect all possible data from the system, and should let the Profiler's feature selector determine which features to consider and which ones to exclude. In our system, we plan to collect information at the application, OS, and network layers and associate all this information within the scope of process, which is the live entity of any computer system. Table 1 illustrates the data we plan to collect per process. In summary, we want to know which processes a user's computer system runs, which network connections it creates and accepts, which files it accesses, whether or not it interacts with a user, and how it utilizes memory. User activities will also depend on the time of the day and will vary on weekdays and weekends. For example, Alice strives for a good work-life balance and does not work on weekends. Bob uses Skype to talk to his son only on Sunday afternoons.

Each row in Table 1 corresponds to the daily activities of one process for a period of 24 hours. *Process ID* (or a hash based on this value) indexes each table entry on a given day. *Parent Process ID* tracks the relationship among threads belonging to the same process. *Active time interval* is a time series corresponding to the time intervals when the process is active. We consider a process active if it is issuing system calls. In our design, if five minutes elapse after the last system call invocation from the process, we close the last period with the end time corresponding to the time of the last system call invocation. The goal is to collect fine-grained information of process activity, which is closely associated with the invocation of system calls. For example, suppose that Alice begins her computer activities at 9 a.m. using Chrome, which starts invoking several system calls until 9:15 a.m. when Alice begins working on a report for her boss using Microsoft Word. At 10:00 a.m., Alice checks her Gmail using Chrome. Because the interval between the two last system-call invocations for Chrome is greater than five minutes (9:15 a.m. and 10:00 a.m.), our extractor ends the last interval at 9:15 a.m. and starts a new time interval at 10:00 a.m.

The *Process Type* represents the process category (*i.e.*, office software, development, browsing, game, computing, *etc.*). *Memory* represents the memory usage for each time interval, and *List of system calls* contains the list of system calls invoked by the processes. Table 2 shows an example of the summarized (i.e., showing only fields with general process information) daily activities for Microsoft Word for a certain weekday. This shows that the user started activity no earlier than 8:30 a.m. and finished activities no later than 6:00 p.m. This line could be part of Bob's profile, but not Alice's, who never starts work before 10 a.m.

The data model also records mouse and keyboard activities in the form of a list of time stamps when the process receives bytes coming from the keyboard or mouse, which indicates direct user interaction. File system information is collected through the *File system features* data structure

Attributes	Process ID (PID)	Parent PID (PPID)	Executable pathname	Active time interval	Process type	Memmory
Description	Integer representing the process ID	Integer representing the parent process ID	The pathname of the process executable file	List of time intervals when the process is active	Integer indicating the process type (game, browsing communication and etc.)	The amount of memory currently held by the process

Attributes (continue)	List of system calls	Mouse activity	Keyboard activity	Filesystem features	Network Features
Description (continue)	List of system calls invoked by the process	List of timestamps representing mouse clicks associated with the process	List of timestamps representing keyboard strokes associated with the process	File feature struct: {File pathname (string), Time accessed (time stamp), Bytes accessed (float)}	Network feature struct: {IP address (Integer, source and dest.), Port Number (Integer, source and dest.), Average bytes upload & download (float), Connection start & end time (time stamp)}

Table 1: Data collection model. Each entry in this table corresponds to the summarized activity for one process in a period of 24 hours.

...	Executable pathname	Time intervals (start/end)			...
...	C:\Windows\ ProgramFiles(x86)\ Microsoft Office 365\ word.exe	8:42 /9.17	...	16:47 /17:50	...

Table 2: Process record sample.

shown in Table 1. For each process, we log a list of records summarizing information about file system access for the process, such as pathname, time of access and average bytes accessed.

The *Network features* field is a list of records summarizing network connection or session activities for the process, such as source and destination IP addresses and ports, average bytes transferred, and start and end time for the connection or session. Table 3 shows the data for one network feature for Chrome. This entry could be part of Alice's profile, but not Bob's, who never connects to 121.294.0.219 (a well-known news site in China).

To collect this data in a cross-layer fashion, we plan to develop a Windows kernel module to perform the extraction. We selected Windows because of its large user base and popularity among malware writers, but the features and the main idea can be implemented in any standard OS. We decided against applying the method of system call hooking for the development of the Extractor because we did not want to override the Windows patch guard module, which protects the system call table from unauthorized modification.

We have been using the *Object Callback* and *Thread/Process CreateNotify* routines for the extraction of general process information. The ProcessCreateNotify routine identifies the current running process, while the Object Callback Routine distinguishes whether the process is active. To record the network features, we have been using the *Windows Filtering Platform Callout* to intercept connections, inbound and outbound Internet traffic. We have been extracting file features using the *Windows Filter Platform Minifilter* routine, which invokes the Extractor every time a file is accessed, allowing the collection of file system-related information.

6. INTELLIGENT PROFILE LEARNING

By combining the different types of activity logs, we will end up with multi-dimensional time series data for each individual. At each time instant t, a particular computer may be executing a certain set of processes, file accesses and network connections.

One possible objective of a security solution employing cross-layer personalization could be the identification of the user or his demographics based on a sample log data for a certain period of time. The demographics categories can be given labels based on gender, age, occupation, race, locality, or even the person's own identity. The successful authentication of a user's identity, however, requires logs of sequential data for a sufficient period of time. If the log granularity is taken to be one full day starting from midnight, then the training data matrix $\mathbf{X}$ is essentially d days of daily log of information for u different users. Thus, there are $u \times d$ rows in $\mathbf{X}$. Concatenating all the features from activity logs can lead to a very large value of the dimensionality p (the number of columns in the data matrix $\mathbf{X}$). Instead of columns, the features could also be multi-dimensional arrays themselves instead of single real-valued entries, which will in turn make the data matrix a multi-dimensional tensor. For the problem of identity recognition, $\mathbf{Y}$ will be the $u \times d$ target vector where each entry is an integer ranging from 1 to u. For example, age classification (old vs. young) would call for a binary valued $\mathbf{Y}$. The problem can also be framed as sequential pattern recognition, where overlapping time windows or periods of log data (*i.e.*, the combined activity from $t - \Delta t$ to t) will act as training observations. If the task of the security solution is to detect anomalies in a personalized fashion, we have to identify the time instants t (or time periods) during which significant deviation is seen from normal or expected behavior.

.....	Executable Pathname	Connection start time	Connection end time	Source IP	Dest IP	Source port	Dest port	.....
.....	C:\Windows\ ProgramFiles(x86)\ Google\Chrome\ Application\Chrome	2016-03-29 10:57:46	2016-03-29 10:58:00	10.255.48.22	121.194.0.239	57323	5355	.....

Table 3: Network record sample.

The problem at hand is not trivial: namely, collecting personalized computer usage data across multiple layers of abstraction on a massive scale and relying on a model to select the right set or projection of discriminating features from such high-dimensional input. Further, it is known in the machine-learning literature that outlier detection is very hard in a high-dimensional space [21] because the data is sparse and the notion of proximity fails to retain its meaning, making every point look like an outlier itself. This requires the application of a robust feature-selection algorithm before building a model for the problem.

The usual approach to deal with high-dimensional data is to apply dimensionality-reduction techniques like Principal Component Analysis (PCA) [22]. For personal profile classification, however, a discriminate feature-selection method like RELIEF [23] can extract those features that can help distinguish the observations better. Further, we can use kernel methods, such as kernel Support Vector Machines (SVMs) [24] to incorporate non-linearity in the feature-projection process. For the problem of malware detection, various time series outlier-detection methods have been outlined in Gupta *et al.* [25]. An anomaly is detected if the distance of the state at a particular time instant from its closest centroid is greater than a threshold. Irrespective of the methods selected, the training has to be done dynamically to be able to adapt to the change of usage behavior over time.

The personalization component of the paradigm can streamline the outlier detection task. For example, if network upload and download size is considered a relevant feature, a download significantly larger than the group average may be considered an anomaly by a conventional statistical anomaly detection-based solution. However, the patterns of such features vary from user to user. For instance, the profile of a data analyst working in a big data company will show a high variance of download and updoad data size, including very large downloads. For an analyst in a government intelligence agency, large downloads and uploads might be atypical and even suspicious.

Another challenge is that the captured logs will have incomplete information. For example, a person might use his computer for a few hours in the morning and then be idle for a few hours because of a meeting or another offline commitment. The missing information in the training data can greatly bias the learning model. Also, the model should take into account benign changes in user behavior, such as changes of occupation or projects, traveling to different time zones, *etc.* Thus, the model should dynamically update itself without explicitly forgetting what has been learned in the past.

Another consideration is how an adversary can interfere with the model and evade profiling. This can be done either by the user themselves to preserve their privacy or by an adversary attempting to evade personalized profiling. We consider this to be the problem of masking or data obfuscation. The statistical model will be confused if the log files contain certain activities that do not characterize the person's normal behavior. An obfuscation component can, for example, ask the computer to perform random activities that do not correspond to the profile, such as opening random files and making network connection at unusual times, with the hopes that this will be incorporated by the model. At first instant, this looks like asking the system to over-perform by running redundant operations at additional computational cost to introduce *noise* into the logs. However, small amounts of random uncorrelated noise (like white Gaussian noise) or outliers will not be able to make the logs anonymous, as modern machine-learning algorithms can provide enough resistance against such evasion techniques. It is a common practice now in computer vision to add salt-and-pepper noise, jitter [26], and occlusion to images while training the model, especially in deep learning, since this helps increase the algorithm's generalization capability. Achieving anonymity, while also reducing the computational burden on the system, can be considered an optimization problem, where we need to add a collaboratively designed sparse-additive outlier into the data matrix such that the eventual classifier will not be able to classify the sample logs. Such constraints may be a simple ℓ_1 norm or any non-convex sparsity enforcing penalty on the additive outlier or noise matrix. Similar approaches have been adopted in the context of analysis of medical data [27].

Statistical deviation from a typical pattern in the sequential data is indicative of possibly malicious activity. This has been the hypothesis behind all statistical anomaly-based Intrusion Detection System (IDS) techniques. However, the distribution that defines normal or usual behavior of networks and software in general is very diverse and not easily explained using limited amount of training data. This is why many anomaly-detection patterns have a high false-positive rate. We hypothesize that a user computer-usage profile may exhibit a more consistent behavior within itself and thus allows for the development of more effective security solutions based on outlier detection.

7. PRELIMINARY RESULTS AND EVALUATION PLANS

We performed preliminary tests using the Firtze Chess Benchmark, which simulates an international chess match. Our test machine had Intel core i7 with Windows 10 installed. We ran each benchmark ten times with and without the current Event Extractor installed, and averaged the results for each case. At least a 1-minute interval was enforced between each test to cool down the CPU. The results are shown in Figure 2. For this benchmark, the performance overhead of the logging extractor was negligible, but we plan

to perform more tests with I/O-intensive benchmarks.

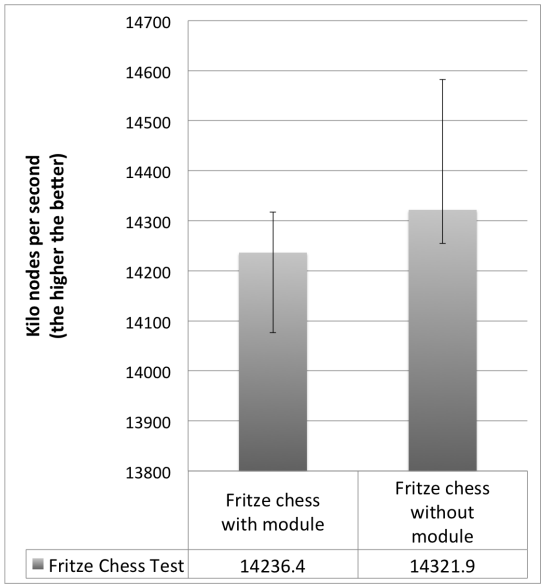

Figure 2: Fritze Chess benchmark results.

We also conducted some preliminary analysis on a dataset our research team obtained in the context of previously collected data in an independent user study [1]. This dataset comprises all the URLs that participants visited over a period of 21 days. The URLs were collected by a browser extension installed in the participants' computers. Participants were 83 Internet users (52 younger; M = 19:9 yrs, SD = 3:40, range: 18–31, 76.9% females; 31 older; M = 70:7 yrs, SD = 6:63, range: 61–88., 51.6% females). Further we had 50.6% (n =42) non-Hispanic White and 33.7% (n = 28) minority participants (African American, n = 7); Hispanic/Latino (n = 7); Asian (n = 12); American Indian/Alaskan Native (n =1); and Other (n=1; n=13 did not report their race)

Each entry in the logs were composed of the URL of the website visited and a time stamp at which the connection was established. We started by distinguishing the web usage behavior of a 25-year-old male graduate student and a 67-year-old retired woman. Figure 3 shows their histogram of web usage over a period of 24 hours. We notice a peak during late night hours for the graduate student, and a peak in the early morning hours for the older woman. These patterns were unique to each of the two participants, while they were equally active during the afternoon hours. These first findings suggest that temporal information regarding computer usage, if selected properly, should be able to distinguish whether a daily log belongs to the graduate student or the old woman. More specifically, SVM with a radial basis function (rbf) kernel achieved up to 93.7% accuracy in classifying these two individuals based on their daily logs.

However, when trying to distinguish between two students from the same university (majoring in similar areas; i.e., engineering and computer science), we were not able to obtain the same accuracy in classification. A two-dimensional PCA projection (*i.e.*, selecting the two most variant directions in the given data matrix) based scatter plot suggested the extent to which these two types of classifications were different. While the data was able to clearly distinguish between the male graduate student and the older woman (see Figure 4), the scatter plot of web usage was inseparable when distinguishing between the two male students from the same university (see Figure 5). This shows that individuals can have similar profiles based on their demographics. However, when we look at the top three websites that the two students visited, the first student had ufl.edu, volleyamerica.com and aol.com as the most frequently visited sites, while for the second student those were ufl.edu, github.com and leetcode.com. These preliminary findings are promising in suggesting that with a large number of features in multiple layers of abstraction we will be able to build a personalized profile for computer users based on their computer activities.

[1]This study was IRB approved.

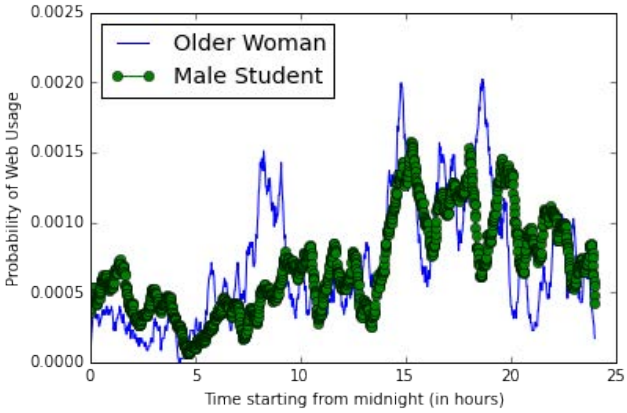

Figure 3: Histograms showing the probability of web usage for a 25-year-old male graduate student (top) and a 67-year-old retired older woman (bottom). The x axis refers to the hours in one full day, *i.e.*, from 0 to 24 hours (midnight).

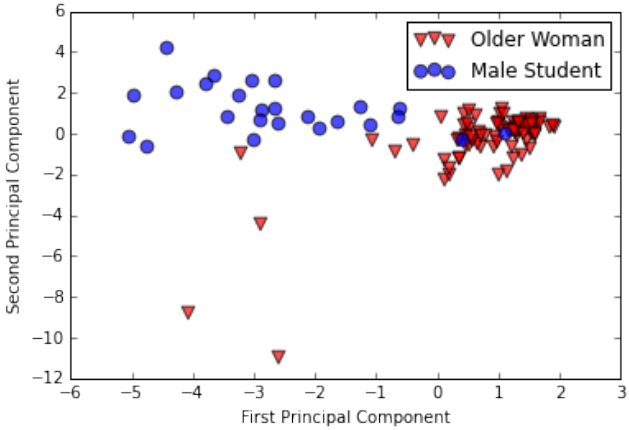

Figure 4: A two-dimensional PCA-projection scatter plot for the web usage of a 25-year-old male graduate student and a 67-old-woman, showing that their web usage times were clearly distinguishable.

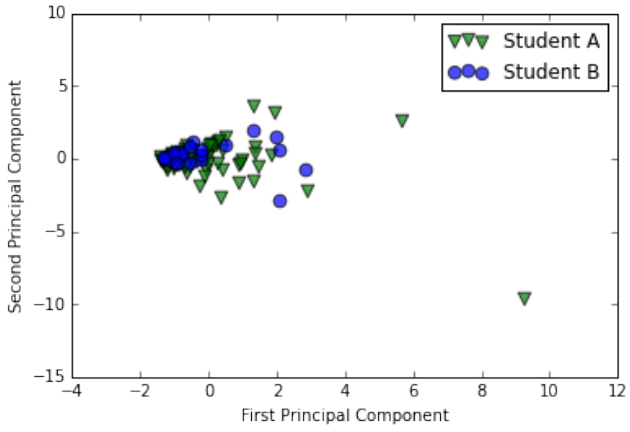

Figure 5: A two-dimensional PCA projection scatter plot for two male graduate students from the same university in similar areas of study (i.e. engineering and computer science), showing that their web usage times were not sufficiently distinguishable.

7.1 Evaluation

Because of the personalization component of this paradigm, an evaluation in the context of a robust user study is warranted. In this section, we discuss one possibility for a field study to evaluate the user acceptability and effectiveness of the paradigm in the context of malware detection and individual profiling.

We envision a field study where the components of the architecture described in Section 4 (extractor and profiler) are either installed in the participant's personal computer or in a study laptop lend to the participant for at least three weeks.

The goal of the study is to validate the hypothesis of uniqueness of the profiles and to discover whether anomaly events, such as malware, produce sufficient perturbations in the profile to be captured by the profiler. Another interesting research question that need to be addressed in the context of a user study, is how well the profiler can adjust the model on legitimate changes of habit, such as a person traveling and changing her time zones, or projects in an organization.Examining this question will require a longer study period to allow for natural changes in the profiles to occur.

The participants' machine in the user study would also have a module that, in the third week of the study, generates events that resemble malware [28] without incurring any actual malicious activity in the participant's computer. A crucial aspect of the study validation will be the recruitment of participants from different demographics to allow for a diverse sample, while making sure that the sample also includes participants from comparable backgrounds(e.g., a group of participants that are comparable in their age range and/or from a similar field of study university students). Classification of participants from a diverse background will allow validation of the uniqueness of the extracted computer system profile. In contrast, classification participants from similar demographics will allow to confirm the hypothesis of group profile.

8. RELATED WORK

This section reviews existing literature addressing personalization or profiling, as well as malware detection.

8.1 Personalization

Song *et al.* [29] provided a Gaussian Mixture Model (GMM) based machine-learning approach for biometric identification of user behavior for authentication. The authentication was based on capturing and estimating the expected number of unique system level events like process creation, registry key changes, and file creation. Yang *et al.* [30] characterized mobile Internet user behavior by capturing HTTP traffic data to cluster users into different groups based on application categories. The goal was to equip network operators with information for resource provisioning. Baglioni *et al.* [31] showed that extracting information from the logs of a particular web server can help identify navigational styles of web users. An interesting pre-processing technique used in this work was extracting semantic information such as the category of the site from the URL name. Nogueira *et al.* [32] classified individuals based on their hourly Internet traffic. A more recent work by Freeman *et al.* [18] added user behavior identification in addition to password authentication based on source IP, geolocation, browser configuration, and time of day. Another interesting application, provided by Duarte *et al.* [19], classified people on the basis of age by determining correlations between chronological age and browsing & search behavior such as ads clicked, click duration and query length.

Another domain where personalization has been researched is the area of creating user-specific recommendations. A comprehensive account of various content-based and collaborative filtering techniques for webpage recommendation was shown in Mobasher [20] and Castellano *et al.* [33]. Chang *et al.* [34] addressed website recommendation using a neural network trained on data gathered from surveys. Davidson *et al.* [35] encouraged client side personalization at the OS level by implementing a model to learn a coarse-grained profile for each person using the Windows phone OS.

Marforio *et al.* [36] suggested personalized security indicators as a phishing-detection solution by experimenting with human subjects on a simulated banking environment. Jiang *et al.* [37] proposed a personalized malware warning system or a "malware recommendation engine," where a risk ranking of each malware sample for each user was computed to determine potential types of malware a particular person may be susceptible to.

In our paradigm, we define the concept of personalization much more broadly, as the core of computer security solutions for computer infrastructure and situational awareness, and to be performed holistically across many layers of abstraction. Barnam *et al.* [38] discussed the issue of monoculture for IT management. They argued that the debate about the monoculture approach of IT management has resurfaced [38]: "It is believed that a collection of identical computing platforms is easier, hence cheaper, to manage because making one set of configuration decisions suffices for all." In spite of these discussions, however, to date, most of the existing IDS systems remain unpersonalized [38].

8.2 Anomalous Activity Detection

The research problem of anomalous activity detection has been well studied under the domain name of IDS which intends to monitor and identify malicious activities. Such systems can be deployed either in the network or at the host site. Broadly, all intrusion-detection systems can be classified into either statistical anomaly-based or signature-based. A specification or signature-based IDS [1–3] compares the current behaviour against a pre-determined database of signatures from known malware. These techniques are accurate, but they can be evaded when attackers use polymorphism and metamorphism to create malware variants; these variants have the same behavior, but different byte signatures. Further, these approaches cannot detect zero-day malware and have a practical detection rate ranging from 25–50% [4].

Statistical or behavioral approaches, in contrast, are based on finding anomalies or deviation from statistically decided normal behaviour. Forrest *et al.* [5, 6] introduced the idea of distinguishing self and other in the context of malware detection. The main idea was that a detector could extract the normal behavior of programs running in a system (self) and thus be able to distinguish malware behavior in the same system (other). Self was extracted based on the set of system calls invoked by the process. The work was groundbreaking and inspired a generation of behavioral-based malware detection solutions [5–12] and automated diversity approaches [39, 40]. This line of work was also based on the insight that computers can behave like the immune system and this unique and diverse architecture can be leveraged for anomaly detection (*e.g.*, [41, 42]).

This previous works characterized normal behavior by sequences of system calls invoked by benign programs. It is based on the hypothesis that malware invokes a different set of system calls than benign software. The approach depends on a training phase that collects sequences of system calls of size N from various benign processes. A detection phase compares the sets of system calls of the process under analysis and raises an alarm if a sequence is not found in the training set. Somayaji *et al.* [6] proposed the design of a malware detection system inspired in the human immune system. Kruegel *et al.* [9] analyzed not only sequences of system calls, but also their arguments. Kirda *et al.* [10] proposed combining static and dynamic analysis for detecting spyware. The authors observed that spyware collected sensitive information from a browser and leaked it. They proposed a detector that identifies browser COM functions and the Windows API calls.

Kolbitsh *et al.* [11] addressed the problem of malware performing system call re-ordering to evade system-call based malware detectors. They analyzed malware in a controlled environment to extract its mission behavior. Then, they extracted program slices responsible for the malware's mission information flows. In a detection phase, the slices were matched against the behavior of an unknown program.

However, system-call-based malware detectors suffer from high positive rates due to the diverse nature of system-calls invoked by applications. This challenge has worsened as programs are becoming increasingly diverse [12,13] and malware more sophisticated [14–16]. Lanzi *et al.* [12] investigated the effectiveness of system-call-based malware detectors on a set of ten hosts and discovered a false-positive rate of approximately 40%. The authors proposed a system-centric approach for the analysis of system calls where they also considered the way programs access OS and its resources. Canali *et al.* [13] investigated the accuracy of system-call-based malware detectors. They analyzed 200 different models with 220 million signatures and proposed a method to measure the quality of a malware detector based on its behavioral model. Many models performed very poorly and the best ones relied on few high-level atoms with their arguments.

Some previous works analyzed the data flow of a program to extract malware behavior. Panorama [43], for example, performed system-level taint-tracking to discover how malware leaks sensitive data. Martignoni *et al.* [44] leveraged hierarchical behavioral graphs to infer high-level behavior of low-level events. The approach traces the execution of a program, performing data-flow analysis to discover relevant actions such as proxying, data leaking and key stroke logging. Ether [45] improved on tracing granularity on single instructions and system calls via hardware virtualization extensions. Ye *et al.* [46] proposed a semi-parametric classification model for combining file content and file relation information to improve the performance of file sample classification. More recently, Bromium [4] proposed the use of virtualization on a per-process basis to isolate every process from the system and from each other.

There are very few publicly available databases to test the efficiency of detection systems. One such is the KDD Cup 99 dataset [47] where the data is the raw binary TCP dump packets. Using the binaries, 41 high-level features under the categories of basic features(e.g. duration, source bytes), content features(e.g. number of root accesses), time-based traffic features (e.g. frequency of connections to same host), and host-based traffic features (e.g. windowed connection rate) were derived. It is a network IDS problem where each connection is labelled as either normal or malicious. Various works like Hoque *et al.* [48] and Wang *et al.* [49] have attempted to do reliable feature selection and anomaly classification on this dataset. The dataset has also been heavily criticized for being outdated and hard to use [50, 51].

An overview of various machine-learning- and data-mining-based anomaly-detection techniques was given by Patcha *et al.* [52]. In order to reduce the false-positive rate in malware determination, an ensemble or multiple classifier system was proposed by Perdisci *et al.* [53]. The data here was again the payload binaries. However, when the data was high-dimensional, such as in our case, outlier detection became extremely difficult, as pointed out by Aggarwal [21].

IDSes have been proposed at the network side [54,55]. McDaniel *et al.* [55] used K-means to analyze historic network communication for hosts within enterprise network and generated malware profiles under different granularities to detect and block worms. Gates *et al.* [56] debated the topic of using the anomaly-detection model for intrusion detection and the validity of various associated assumptions like sparsity and anomalous nature of the attacks. But their research mainly focused on network traffic and suffered from the loss of error.

Sommer *et al.* [57] discuss several challenges that come up while using machine learning for intrusion detection. Our approach of personalized cross-layer will still be susceptible to these concerns because it is based on the same anomaly-detection premise. However, the use of cross-layer training data will be more holistic, which can improve the detector

effectiveness. Also, a personalized detection shall takes into account the cyber usage for a single person, which is much more coherent and informative of normal behavior of a user than across-participant training data that is characterized by high variability.

For our paradigm, OS & software updates and user behavior changes will present challenges. The implementation of the paradigm will need to be aware of the start of updates and deploy dynamic thresholds when a change of pattern is observed. The implementation can leverage user feedback to distinguish user behavior change from anomalous attack.

The importance of addressing malicious activity from a cross-layered perspective was also inspired by Crandall *et al.* [58] and Oliveira *et al.* [59] in their discussion of how software vulnerabilities cross layers of abstraction. Many vulnerabilities manifest at several layers of abstraction and solutions that focus on a single layer are easily bypassed by a clever attacker who understands the vulnerability well. For example, at the application layer buffer overflows are caused by an input string that exceeds the bounds of an array. At the OS layer, it is caused by the way programs are laid out into the stack with control data in band with data.

Our paradigm has the potential to boost the effectiveness of behavior-based detectors by reducing false-positive and false-negative rates that are caused by the diversity of computer system usage. For example, a connection to a Russian domain from a computer of a Russian descent should probably be labeled as typical, while such connection coming from a computer of a user computer with no ties to Russia should be considered suspicious and flagged. Another example is an organization's computer making a great number of network connections in a short period of time. If this behavior deviates from the user profile, there is a great chance that this computer has been compromised and used as part of a DoS attack. However, if the computer belongs to an employee whose primary task is to perform stress tests in a web server, the behavior is typical and legitimate. A one-sizes-fits-all anomaly detector, which focus on general or per-malware category rules and operating at a single layer of abstraction, will usually err in these situations.

9. SUMMARY OF DISCUSSION

The lively discussion at the workshop focused mainly on five topics: (i) feature selection, (ii) profile stability, (iii) privacy, (iv) responsibility and (v) user studies.

Feature selection. This was the most discussed topic at the workshop. The attendees agreed that it will be a critical part of our research. Tom Longstaff pointed out that the set of useful features will be actually a small fraction of all the features we will collect. The attendees believed that the high-dimensional and temporal data set will become one of the main challenges for the machine-learning classifier.

Further, the feature selection process will greatly depend on both our target and the user. This selection will not be completely stable. Paul van Oorschot suggested that the features needed for detection and prevention could be very different. We also believe that the best selection of features might vary per user but we still need the data from a user study to test our hypothesis.

Deborah Shands suggested that a large set of features will enable the development of unlimited feature sets with distinct features coming from multiple layers. Different feature sets may be needed for different tasks to achieve maximum performance. Machine-learning methods with built-in feature selection, such as deep-learning, are promising to address these issues, but further research is need to determine the effectiveness of these solutions.

Profile stability. Marco Carvalho raised the question of profile stability, or how our proposed approach will withstand benign changes in the profile caused by travel, project changes, etc. We are cognizant of the problem and plan to apply RNN, which is a category of deep-learning algorithm that takes small windows of data as input. Analysis of small windows of input could allow us to make decisions as close to real time as possible. However, we still need to prove RNN's efficiency in this context. Additionally, our assumption is that the protection target are enterprises where employees' profiles are relatively stable. We also envision leveraging user-feedback through an interface to inform the model about significant profiles changes. Marco Carvalho also suggested the use of mixture modeling to address the this question.

Privacy. Many attendees raised the concerns regarding privacy and cross-country differences of the definition and expectations of privacy. Although we have not verified it is possible, we expect that the proposed approach can be fine-tuned to operate according to the level of privacy required in a particular country or organization. For example, one possible but not verified solution is to use a different set of data depending on the laws of the country.

Responsibility. The responsibility of handling an alarm, raised by Hilary Hosmer, is another interesting issue. However, our research on the practical division of responsibility under such case is limited. Currently, we expect the alarm to be handled by either the employee or a system administrator, in line with the organization's policies. But the solution has not shown yet to be practical or effective with empirical data, and is likely to change based on results of future research.

Evaluation through a user study. As part of this research, we will collect profiles from real Internet users in an IRB-approved user study involving participants from the Gainesville, FL area. This data will allow us to test our hypothesis about the uniqueness of extracted profiles and to determine the extent to which outlier events in the profile can be distinguished. The number of participants for the user study will be determined via statistical power analysis method [60].

10. CONCLUSIONS

Today's level of sophistication and stealthiness of attacks in computer infrastructures of organizations and government agencies indicates that our cyber security solutions are outdated. While adversaries are designing attacks that are targeted, our solutions still operate in a "*one-size-fits-all*" fashion. To counter this "new normal," we advocate a novel paradigm where cross-layer personalization is a premier component in the design of security solutions for situational awareness and protection of computer infrastructure. The paradigm revolves around the observation that computers have been used in a personalized fashion, with no or very little device sharing, and that individual profiles can be used to build a signature for a user. We proposed that personalization should encompass all layers that make up the computer system: application, OS, and network, so that different data can be correlated in a robust profile.

Such cross-layer personalization should be the foundation for the next-generation malware, APT, insider-attack detection, and for continuous user authentication. We presented the initial design and preliminary results of the proposed paradigm, which showed promise of the cross-layer personalization approach. There are also intrinsic security and privacy challenges in collecting computer behavioral data from citizens' devices. We hope the cross-layer personalization paradigm will motivate new research on malware & insider attack detection, and continuous authentication.

Acknowledgments

We would like to thank our shepherd Mohammad Mannan and Paul Van Oorschot for guidance in writing the pre-proceeding and post-proceeding version of the paper, and the NSPW 2016 anonymous reviewers for their valuable feedback. This research has been supported by DARPA Trusted Computing Project, grant No. FA8650-15-C-7565, and MIT Lincoln Laboratory through Air Force Contract No. A8721-05-C-0002 and/or FA8702-15-D-0001.

11. REFERENCES

[1] S. Kumar and E. H. Spafford, "An application of pattern matching in intrusion detection," tech. rep., Purdue University, July. 1994.

[2] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State transition analysis: A rule-based intrusion detection approach," *IEEE Trans. Softw. Eng.*, vol. 21, pp. 181–199, Mar. 1995.

[3] G. Vigna and R. A. Kemmerer, "Netstat: a network-based intrusion detection approach," in *Proceedings of the 14th Annual Computer Security Applications Conference*, pp. 25–34, ACM, 1998.

[4] "Bromium end point protection (https://www.bromium.com/)," Apr. 2016.

[5] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff, "A sense of self for unix processes," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 120–128, IEEE, 1996.

[6] A. Somayaji, S. Hofmeyr, and S. Forrest, "Principles of a computer immune system," in *Proceedings of the 1997 Workshop on New Security Paradigms*, NSPW '97, (New York, NY, USA), pp. 75–82, ACM, 1997.

[7] S. A. Hofmeyr, S. Forrest, and A. Somayaji, "Intrusion detection using sequences of system calls," *Journal of Computer Security*, vol. 6, pp. 151–180, Aug. 1998.

[8] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls: Alternative data models," *Proceedings of IEEE Symposium on Security and Privacy*, pp. 133–145, 1999.

[9] C. Kruegel, D. Mutz, F. Valeur, and G. Vigna, "On the detection of anomalous system call arguments," *ESORICS*, vol. 2808, pp. 326–343, 2003.

[10] E. Kirda, C. Kruegel, G. Banks, G. Vigna, and R. A. Kemmerer, "Behavior-based spyware detection," in *Proceedings of the 15th USENIX Security Symposium*, USENIX-SS'06, 2006.

[11] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang, "Effective and efficient malware detection at the end host," in *Proceedings of the 18th USENIX Security Symposium*, USENIX-SS'09, pp. 351–366, 2009.

[12] A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "Accessminer: Using system-centric models for malware protection," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, CCS '10, pp. 399–412, 2010.

[13] D. Canali, A. Lanzi, D. Balzarotti, C. Kruegel, M. Christodorescu, and E. Kirda, "A quantitative study of accuracy in system call-based malware detection," in *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, ISSTA 2012, pp. 122–132, 2012.

[14] T. Wrightson, *Advanced Persistent Threat Hacking: The Art and Science of Hacking Any Organization*. McGraw-Hill Education, 1st ed., 2014.

[15] "Email Attacks: This Time It's Personal (http://itknowledgeexchange.techtarget.com/security-detail/cisco-report-email-attacks-this-time-its-personal/) ," Jul. 2011.

[16] "RSA: SecurID Attack Was Phishing Via an Excel Spreadsheet (https://threatpost.com/rsa-securid-attack-was-phishing-excel-spreadsheet-040111/75099/) ," Apr. 2011.

[17] K. Kupferschmiddt, "A Trail of Microbes - The Unique Mix of Bacteria You Leave Behind Wherever You Go Might Be Used to Identify You," *Science*, vol. 351, no. 6278, 2016.

[18] D. M. Freeman, S. Jain, M. Dürmuth, B. Biggio, and G. Giacinto, "Who are you? a statistical approach to measuring user authenticity," in *23rd Annual Network & Distributed System Security Symposium (NDSS). The Internet Society*, 2016.

[19] S. Duarte Torres, I. Weber, and D. Hiemstra, "Analysis of search and browsing behavior of young users on the web," *ACM Transactions on the Web (TWEB)*, vol. 8, no. 2, p. 7, 2014.

[20] B. Mobasher, "Data mining for web personalization," in *The adaptive web*, pp. 90–135, Springer, 2007.

[21] C. C. Aggarwal and P. S. Yu, "Outlier detection for high dimensional data," in *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, SIGMOD '01, (New York, NY, USA), pp. 37–46, ACM, 2001.

[22] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2002.

[23] K. Kira and L. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *Tenth National Conference on Artificial Intelligence (AAAI-92)*, pp. 129–134, MIT Press, 1992.

[24] B. Schoelkopf and A. J. Smola, *Learning with Kernels*. Cambridge, MA: The MIT Press, 2002.

[25] M. Gupta, J. Gao, C. Aggarwal, and J. Han, "Outlier detection for temporal data," *Synthesis Lectures on Data Mining and Knowledge Discovery*, vol. 5, no. 1, pp. 1–129, 2014.

[26] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[27] G. Szarvas, R. Farkas, and R. Busa-Fekete, "State-of-the-art anonymization of medical records using an iterative machine learning framework," *Journal of the American Medical Informatics Association*, vol. 14, no. 5, pp. 574–580, 2007.

[28] A. Gregio, R. Bonacin, A. C. de Marchi, O. F. Nabuco, and P. L. de Geus, "An ontology of suspicious software behavior," *Applied Ontology*, vol. 1, pp. 1–21, 2016.

[29] Y. Song, M. Ben Salem, S. Hershkop, and S. J. Stolfo, "System level user behavior biometrics using fisher features and gaussian mixture models," in *Security and Privacy Workshops (SPW)*, pp. 52–59, IEEE, 2013.

[30] J. Yang, Y. Qiao, X. Zhang, H. He, F. Liu, and G. Cheng, "Characterizing user behavior in mobile internet," *Emerging Topics in Computing, IEEE Transactions on*, vol. 3, no. 1, pp. 95–106, 2015.

[31] M. Baglioni, U. Ferrara, A. Romei, S. Ruggieri, and F. Turini, "Preprocessing and mining web log data for web personalization," in *AI* IA 2003: Advances in Artificial Intelligence*, pp. 237–249, Springer, 2003.

[32] A. Nogueira, M. R. De Oliveira, P. Salvador, R. Valadas, and A. Pacheco, "Classification of internet users using discriminant analysis and neural networks," in *Next Generation Internet Networks*, pp. 341–348, IEEE, 2005.

[33] G. Castellano, L. C. Jain, and A. M. Fanelli, *Web Personalization in Intelligent Environments*. Springer Publishing Company, Incorporated, 1st ed., 2009.

[34] C. C. Chang, P.-L. Chen, F.-R. Chiu, and Y.-K. Chen, "Application of neural networks and Kanos method to content recommendation in web personalization," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5310–5316, 2009.

[35] D. Davidson, M. Fredrikson, and B. Livshits, "Morepriv: Mobile os support for application personalization and privacy," in *Proceedings of the 30th Annual Computer Security Applications Conference*, pp. 236–245, ACM, 2014.

[36] C. Marforio, R. J. Masti, C. Soriente, K. Kostiainen, and S. Capkun, "Personalized security indicators to detect application phishing attacks in mobile platforms," *CoRR*, vol. abs/1502.06824, 2015.

[37] J.-Y. Jiang, C.-L. Li, C.-P. Yang, and C.-T. Su, "Poster: Scanning-free personalized malware warning system by learning implicit feedback from detection logs," in *Proceedings of the 21st ACM Conference on Computer and Communications Security*, CCS '14, pp. 1436–1438, ACM, 2014.

[38] D. Barman, J. Chandrashekar, N. Taft, M. Faloutsos, L. Huang, and F. Giroire, "Impact of IT monoculture on behavioral end host intrusion detection," in *Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, WREN '09, (New York, NY, USA), pp. 27–36, ACM, 2009.

[39] S. Forrest, A. Somayaji, and D. Ackley, "Building diverse computer systems," in *Proceedings of the 6th Workshop on Hot Topics in Operating Systems (HotOS-VI)*, HOTOS '97, (Washington, DC, USA), pp. 67–72, IEEE Computer Society, 1997.

[40] E. G. Barrantes, D. H. Ackley, S. Forrest, and D. Stefanović, "Randomized instruction set emulation," *ACM Trans. Inf. Syst. Secur.*, vol. 8, pp. 3–40, Feb. 2005.

[41] S. Forrest, S. A. Hofmeyr, and A. Somayaji, "Computer immunology," *Commun. ACM*, vol. 40, pp. 88–96, Oct. 1997.

[42] S. A. Hofmeyr and S. A. Forrest, "Architecture for an artificial immune system," *Evol. Comput.*, vol. 8, pp. 443–473, Dec. 2000.

[43] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda, "Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis," *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 116–127, 2007.

[44] L. Martignoni, E. Stinson, M. Fredrikson, S. Jha, and J. C. Mitchell, "A layered architecture for detecting malicious behaviors," in *Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection*, RAID '08, pp. 78–97, 2008.

[45] A. Dinaburg, P. Royal, M. Sharif, and W. Lee, "Ether: Malware analysis via hardware virtualization extensions," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, CCS '08, pp. 51–62, ACM, 2008.

[46] Y. Ye, T. Li, S. Zhu, W. Zhuang, E. Tas, U. Gupta, and M. Abdulhayoglu, "Combining file content and file relations for cloud based malware detection," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, (New York, NY, USA), pp. 222–230, ACM, 2011.

[47] S. J. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. K. Chan, "Cost-based modeling for fraud and intrusion detection: Results from the jam project," in *DARPA Information Survivability Conference and Exposition. DISCEX'00. Proceedings*, vol. 2, pp. 130–144, IEEE, 2000.

[48] M. S. Hoque, M. Mukit, M. Bikas, A. Naser, *et al.*, "An implementation of intrusion detection system using genetic algorithm," *arXiv preprint arXiv:1204.1336*, 2012.

[49] K. Wang and S. J. Stolfo, "Anomalous payload-based network intrusion detection," in *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection*, RAID '08, pp. 203–222, 2004.

[50] T. Brugger, "KDD Cup '99 dataset (Network Intrusion) considered harmful." KDnuggets News, n18: item4, 15 Sep 2007.

[51] V. Engen, J. Vincent, and K. Phalp, "Exploring discrepancies in findings obtained with the kdd cup '99 data set," *Intell. Data Anal.*, vol. 15, pp. 251–276, Apr. 2011.

[52] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Journal of Computer Networks*, vol. 51, no. 12, pp. 3448–3470, 2007.

[53] R. Perdisci, D. Ariu, P. Fogla, G. Giacinto, and W. Lee, "Mcpad: A multiple classifier system for accurate payload-based anomaly detection," *Journal of*

Computer Networks, vol. 53, no. 6, pp. 864–881, 2009.

[54] S. Staniford-Chen, S. Cheung, R. Crawford, M. Dilger, J. Frank, J. Hoagland, K. Levitt, C. Wee, R. Yip, and D. Zerkle, "Grids – a graph based intrusion detection system for large networks," in *In Proceedings of the 19th National Information System Security Conference*, pp. 361–370, 1996.

[55] P. D. McDaniel, S. Sen, O. Spatscheck, J. E. van der Merwe, W. Aiello, and C. R. Kalmanek, "Enterprise security: A community of interest based approach.," in *13th Annual Network & Distributed System Security Symposium (NDSS)*, pp. 1–3, 2006.

[56] C. Gates and C. Taylor, "Challenging the anomaly detection paradigm: a provocative discussion," in *Proceedings of the 2006 Workshop on New Security Paradigms*, pp. 21–29, ACM, 2006.

[57] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proceedings of IEEE Symposium on Security and Privacy*, pp. 305–316, IEEE, 2010.

[58] J. R. Crandall and D. Oliveira, "Holographic vulnerability studies: Vulnerabilities as fractures in interpretation as information flows across abstraction boundaries," in *Proceedings of the 2012 Workshop on New Security Paradigms*, NSPW '12, (New York, NY, USA), pp. 141–152, ACM, 2012.

[59] D. Oliveira, J. Crandall, H. Kalodner, N. Morin, M. Maher, J. Navarro, and F. Emiliano, *An Information Flow-Based Taxonomy to Understand the Nature of Software Vulnerabilities*, pp. 227–242. Cham: Springer International Publishing, 2016.

[60] P. A. Lachenbruch, "Statistical power analysis for the behavioral sciences," *Journal of the American Statistical Association*, vol. 84, no. 408, pp. 1096–1097, 1989.