A Socially-Aware Operating System for Trustworthy Computing

Daniela Oliveira, Dhiraj Murthy
Bowdoin College
{doliveir, dmurthy}@bowdoin.edu

Henric Johnson
Blekinge Institute of Technology
henric.johnson@bth.se

S. Felix Wu, Roozbeh Nia, Jeff Rowe
University of California Davis
{wu, rowe, nia}@cs.ucdavis.edu

***Abstract*—Traditional security models based on distinguishing trusted from untrusted pieces of data and program behavior continue to face difficulties keeping up with attackers levels of sophistication and ingenuity. In this position paper, we present a novel computing paradigm for trustworthy computing whose application, operating system (OS) and architecture can leverage social trust to enhance the robustness and diversity of security mechanisms of any Internet-based computing environment. Our model would allow online social network (OSN) users to assign trust values to her friends in a privacy-preserving fashion and maintain a trust repository with trust values for objects like URLs, email addresses, IP addresses and other pieces of data that can be consumed by a socially-aware OS, allowing for fine-grained trust decisions that take into account user context and add diversity to host behavior. Our model also automatically infer trust values for people a user is not directly connected. In this paper we sketch the design of a socially-aware operating system kernel and identify several research challenges for this new paradigm.**

I. Introduction

In the last few years we have observed two key trends. First is the rise in popularity of OSNs [1], which allow human social relationships to be captured digitally and utilized by many different online applications. In the virtual world users interact with people and entities using almost the same protocols from real life, which are based on trust and reputation. On-line users establish virtual networks of friends, communicate with them and share media, photos, status and messages. OSNs are based on a capital that has not yet been fully leveraged in computer security: social trust.

The second trend is the increase in complexity of malware. The current generation of attackers is extremely creative, financially and politically motivated, and structured much like any well-operated criminal organization. Traditional security models and solutions have difficulty keeping up with these attackers' level of innovation and ingenuity.

Given these two trends (Figure 1), we have been developing a novel trustworthy computing paradigm that adds the role of the human user and is based on social trust. We believe that incorporating the characteristics of human users can make a significant improvement in cyber security. One particular challenge is that the widespread lack of user knowledge about the technical issues causes them to make poor decisions, for which software solutions are not very helpful. We believe that social informatics of this sort can bridge this gap by providing good collaborative models that include social trust that allow

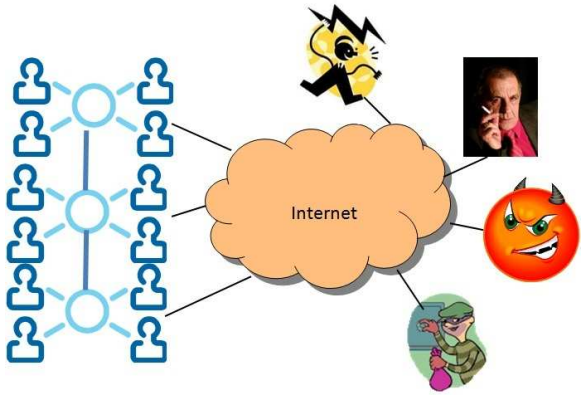

Fig. 1. Two trends: popularity of OSNs and complexity of malware landscape.

users to defend themselves more effectively against malware attacks. To this end, we believe that the OS should not merely manage processes, memory, and I/O devices syntactically. Applications and the OS should become socially-aware and leverage social trust to enhance the robustness and diversity of their security mechanisms. Our model would allow OSN users to assign trust values to other users and pieces of data (software programs, URLs, files, e-mail addresses) that can be consumed by a socially-aware OS and architecture. This would allow fine-grained trust decisions about data that take into account user context, thus adding sophistication and diversity to a host's behavior.

Nearly all current security models work by distinguishing trusted from untrusted pieces of data and program behaviors. These models (which include signature-based, behavior-based and information flow tracing-based solutions) are automated, rigid, and threat-specific. Thereby, they are not keeping up with the new generation of malware and attackers. Moreover, with the shift to a Web-based computing paradigm [2], the task of distinguishing trusted from untrusted information automatically (usually by assuming network data is untrusted) becomes more complex.

Our model leverages social trust to distinguish a continuum of trusted/untrusted values, thus adding flexibility to security solutions. This addition is key to thwarting attacker strategies by making it harder to predict a host behavior in the face of a particular threat. Our model can also be used to (i) implement stronger security policies that take into account user context and a continuum of trust values for heterogeneous objects, (ii)

better detect and filter spam e-mails, scams and phishing by employing both the social path between sender and receiver and the implicit reputation of a sender to determine trustworthiness for a message, and (iii) streamline the decision-making process of installing a new software.

II. Limitations of Traditional Prevention and Detection Models

Current malware detection and prevention solutions usually work by distinguishing malicious and benign pieces of data, program behaviors, and information-flow traces. These approaches are much automated, target a particular type of threat, and lack diversity. In this section we identify the three main categories of such solutions and describe why they are not keeping up with the current generation of malware and attackers.

A. Signature-based solutions

This type of approach identifies malware and attacks by using patterns that are characteristic of a particular exploit or family of malware [3]. Although still the predominant approach to identify malware, it can be defeated by code obfuscation, polymorphism (ciphers malware code to avoid detection) and metamorphism (changes malware body without changing its functionality). Further it cannot prevent zero-day attacks, which do not have a signature previously generated.

B. Behavior-based solutions

These solutions try to address the shortcomings of signature-based approaches by focusing on the dynamic behavior of a program [4]–[7]. They monitor system activity and classify it as either normal (benign) or anomalous (malicious) by using either precise specifications of malware behavior or heuristics. The idea is that program behavior will remain the same in spite of its polymorphic variants [4]. In order to determine what is anomalous or benign, the system must be trained to recognize normal system activity. This is usually accomplished with artificial intelligence techniques. Although this approach can deal better with malware variants compared to signature-based approaches, it is susceptible to false positives. Moreover, recent research [8] highlights other challenges for these approaches, such as lack of training data and enormous variability in input data, and suggests that such machine learning techniques are not appropriate for intrusion detection systems (IDSs).

Research efforts have also focused on describing malware in terms of violations to an information-flow policy, which defines how information moves in a system [9]–[12]. The main idea of dynamic information flow tracking (DIFT) systems is to tag data from a particular source (e.g, network) with some extra information and track how this data flows throughout the system. This extra information can be just a bit or a pointer to a memory area containing more complex data [13]. Such information flow tracing has been widely used in security solutions, especially to detect control-flow hijacking attacks based on memory corruption. In this type of application when tainted bytes (coming from network) are used in control-flow instructions (such as jump) or overwrite control data (such as PC register), the system interprets it as a violation. Information flow tracing carries considerable performance overhead, which is a problem addressed by recent work [14]–[16]. Another challenge which has started to be addressed is stability [13], which prevents these systems from tracking all necessary dependencies [9]. Moreover, current DIFT-based IDSes usually support only two levels of trustiness (trusted or untrusted). Having only two levels of trustiness fails to encompass other nuances in the concept of trust, given the current paradigm shift in which computer users meet most of their computing needs through a web browser.

C. What is missing?

Although the categories of solutions described above are indispensible and proved very successful for particular types of threats, their computing paradigm is not strong enough to defeat the generation of malware and attackers we are currently facing. The software diversity tactics employed by malware writers require new detection techniques employing a novel factor that could introduce unpredictability on how a piece of malicious code will behave on a certain host. Diversity is key as it increases the cost for attackers and decreases their chances of success. We need a novel way to distinguish trusted/untrusted data, program behavior, URLs, files and also recognize that we are dealing with a continuum of values between totally trusted and totally untrusted. We believe that human collaboration and their intrinsically unique definition of trust is key to move towards a new socially-aware paradigm for trustworthy computing.

III. The Challenges of Computing with Social Trust

Trust is a cognitive process that discriminates people, objects and entities that are trustworthy, distrusted or unknown [17]. According to Georg Simmel, people choose whom they will trust in which respects and under which circumstances based on good reasons. He also states that the degree of cognitive familiarity with the object of trust is a continuum with the extremes being total knowledge and total ignorance. In Sociology, trust is viewed as an essential commodity, a functional prerequisite for society, whose alternatives are chaos and fear [18]. It is so indispensable that allows us to accept its unavoidable element of risk and uncertainty. Social relations are based on trusted relationships and you trust a person (or an entity) because “knowing what you know of hers disposition, available options and their consequences, you expect that she will choose to take an honest course of action” [19]. Trust is based on reputation, which is a capital asset in which people may invest greatly and which is acquired slowly but can be destroyed very quickly [19]. As described by Marsh and Biggs [20] the acceptance of using social trust as a tool for making decisions about trustworthiness implies risk as trust can sometimes get misplaced. In our case we see the employment of trust as a bonus as our socially-aware system will not be less secure than it would be if we have

not employed trust. Without our socially-aware paradigm the system behaves in its standard fashion: the OS does not enforce any restrictions on activities originated from networked data. On the other hand, DIFT-based systems enforce, in general, restrictive policies: all networked data is suspicious and their activities should be restricted. We propose a model to represent a continuum between totally trusted and totally unstrusted by leveraging social trust.

Computing with social trust is a relatively new research area but it has been receiving an increasingly amount of attention from the research community, which seeks to develop models and systems capable of defining, modeling and employing social trust in applications [21]–[26].

Golbeck [24] studied the problem of utilizing the structure of an OSN and the trust relationships within to infer how much two people that are not directly connected trust one another and to integrate it in applications. She considers only networks such as LinkedIn where individuals that are directly connected explicitly assign trust to one another in a binary scale and infers trust relationships from individuals that are not directed connected. Her model does not consider a trust repository and the inference of trust among its objects.

Richardson et al. [27], introduce a mathematical and a probabilistic model to infer trust and belief in statements made by users in the Web. Their model is general in the sense that they do not specifically consider a social network, but a system where users explicitly assign trust values to other users and statements these users make in this system. They run their experiments in Epinions, a user-oriented product review website where users specifically specify which users and statements they trust and use this model to order the product reviews seen by each person. Guha et al. [28] also used Epinion to study whether distrust can be propagated and inferred like trust by converting ratings to binary values representing trust and distrust.

IV. The Socially-Aware Framework

A. The OS Kernel

We envision an architecture with the following components (Figure 2): (i) an extended Facebook-like OSN infrastructure that maintains not only user profiles and their social networks, but also an optional trust repository per user, (ii) users that can assign trust values to their friends (or have these values inferred through machine learning techniques [29], [30]) in a privacy-preserving fashion and feed their trust repository with trust values for objects such as URLs, IP addresses, files, e-mail addresses, and (iii) a socially-aware OS capable of retrieving and managing trust repositories locally and exporting a trust interface to user applications.

These applications retrieve information (through the OS trust interface) from a user trust repository to make better decisions about the security or trust level of the objects they access. The socially-aware OS also leverages the trust repository data to employ strong and fine-grained security policies about processes, files, modules in the systems and to propagate trust values in collaboration with the architecture layer through a virtual machine (VM).

B. Usage Model

The main purpose of a traditional OS kernel is to manage/allocate various resources (such as memory, files and I/O devices) and to provide abstractions to heterogeneous lower-level resources (e.g., a v-node) for all the active processes. The key difference between our socially-aware OS and a traditional one is that, in order to fully support this novel computing paradigm, we treat social informatics as a type of first-class resource. Furthermore, each of the traditional resources, such as file system, processes and memory will be augmented with a trust value. As an example, when a user process is accessing a particular file, the social context may be a trust value taken from [0,1].

In this model, a user first logs into the OSN that maintains her trust repository and through its extended interface synchronizes her trust repository with the OS. This synchronization is part of the OS trust interface and is implemented as a system call. We also enhance the OS with socially-aware versions of selected syscalls. The trust repository contains trust values for objects of several types and the OS will use this information to make the best possible decisions regarding execution of code or access to system resources. Moreover, user applications can leverage the OS trust interface to get information from a user repository to enforce their own security policies. For example, when a user attempts to save a file into the file system coming from a URL, the browser can use the OS trust interface to propagate trust values for this file. The trust-aware version of the system call *sys_write* can take a URL as a parameter and if there a trust value assigned for that particular URL root domain in the OS cached trust repository, the file inode is updated with this trust value. If the file being saved cannot be associated with any object in the user trust repository, it is tagged as untrusted (trust value 0) in the system.

A user can also update her trust repository with trust values for IP addresses. For example, user Alice can trust network packets arriving from machines whose IP addresses belong to her university. Thus, when a network packet arrives into the system, the OS checks if there is a trust value assigned for the source IP address. If there is, the packet bytes can inherit at the architecture level (VM) the trust level of the source IP address. An email application and browser can use the OS trust interface in a similar way to better filter spam e-mail and URL content, respectively. For example, an e-mail message will inherit the trust level of the sender. The e-mail application retrieves the trust level of the sender and can decide how to treat the message.

C. Assumptions

We have the concept of establishment time: immediately after the boot sequence all files, OS code, processes, memory areas in the host are considered trustworthy. In this case there are no trust information tagged to the objects in the system and the absence of tagging information means that the object is local and trusted. After the establishment time, every byte arriving in the system from the Ethernet device and USB devices is considered as untrusted unless there is a trust value specified for the source IP address. We have adopted

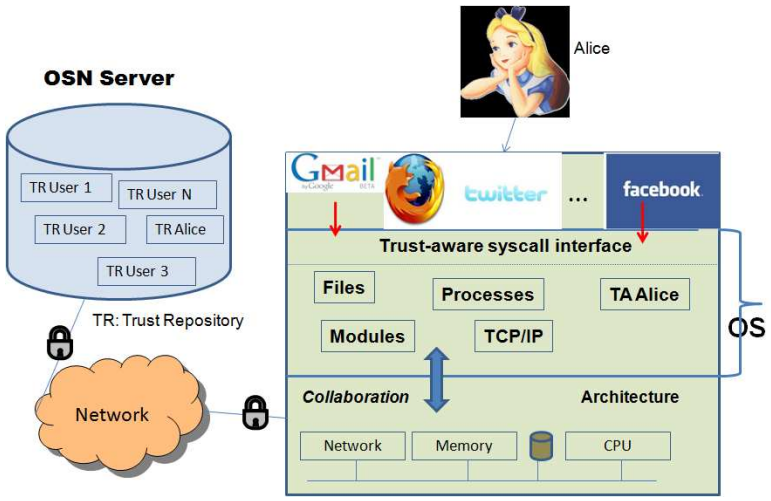

Fig. 2. The Socially-Aware Architecture.

this definition of trust because the network and flash drives are the main attack vectors for several types of malware and exploits. We recognize that this assumption does not come without limitations. For instance, a host could be victim of an internal attack, some piece of hardware could be malicious or the OS could be already compromised at boot time. However we still believe that such assumption could provide coverage for a great number of attacks. We plan to adopt the same DIFT system we introduced in [31].

D. Modeling and Inferring Trust for Users and Objects

As we have discussed before, our extension to a Facebook-like OSN infrastructure allows a user to maintain a trust repository associated with her account, where the user can associate trust values to all her friends and objects like URLs, IP addresses, software, patches, and games. Friends of a user are represented by e-mail addresses and these addresses are inserted automatically in the trust repository if configured as a sharable piece of information in the friend profile. If the user does not specify a trust value for a friend, we consider the friend trusted (trust value 1). Other objects like URLs and IP addresses must be manually included by the user in the trust repository. If no trust value is included for these values, we consider them as trusted. A software, patch, game or any other object can be represented by a URL where they can be downloaded and an attribute (software, patch, or file).

Suppose a user logged into an OS wants to leverage the socially-aware paradigm for trustworthy computing. She will open a browser, log into her Facebook-like OSN and invoke the synchronization operation. This operation will invoke a new system call into the OS (*social_synch*) which initiates a secure SSL/TCP connection with the OSN to retrieve a user merged trust repository. In this model, trust relationships will be shared in a privacy-preserving fashion. For example, a user can only leverage a friend's e-mail address in a trust repository if the friend has explicitly specified that such information is sharable in his/her OSN profile. If a piece of information is not visible, it will not be exported.

The user merged trust repository contains the trust values assigned by this user (or inferred through machine learning [29], [30])and also the inferred trust values for friends of friends up to a distance k from the original user and also inferred trust values for objects in these users trust repositories. For example if $k = 2$, we will be working on the graph composed by the user, her friends and friends of her friends. It is important to point out that any user can define the level of sharing of her trust repository (no one, friends, friends of friends, everybody). As a result there is no privacy violation as the user defines what can be shared and how it will be shared. By default, all objects in a user trust repository are private and cannot be shared.

To model and infer trust, we adapt the trust model by Richardson et al. [27] due to the simplicity of its assumptions and its generality. It is a mathematical and probabilistic model to infer trust in users and belief in statements made by them in the Web. Their model is general in the sense that they do not specifically consider a social network, but a system where users explicitly assign trust values to other users and statements these users make in this system. We start with initial assigned trust values t_{ij}, where user i is friends with user j. Then we infer how much a user trusts another user in the network even if she is not directly connected to them (friend relationship).

Our adaptation of the model is as follows. If user i is friends with user j, this means that i trusts j by an amount specified by t_{ij}. Also, if user j is friends with user k, then j has some trust t_{jk} in user k and, then user i should have some trust t_{ik} in user k, which is a function of t_{ij} and t_{jk}. We assume a network of N users. The extended Facebook-like OSN infrastructure starts with a NxN matrix T, called the personal trust values matrix, where t_{ij} contains the trust user i has on user j he/she befriends. These values are explicitly assigned by the users. In this matrix t_{ij} is not necessarily equal to t_{ji}, and t_i represents the row vector of user i trust in other users. Thus, in this matrix, t_{ik} represents how much user i trusts user k and t_{kj} represents how much user k trusts user j, and $(t_{ik}.t_{kj})$ represents the amount user i trusts user j via

k. The amount that user i trusts user j via any single other node is thus given by $\sum_k(t_{ik}.t_{kj})$. Then we compute for any user i her trust on any user j in the network.

The trust between any two users is given by a trust matrix $\mathcal{T}$ (merged trusts matrix) that represents the merged trusts on the same graph where there is a path between user i and j if i is friends with j. We infer trust values between any user i and j, independently if i is friends with j using an aggregation function which concatenates trusts along every path between them by applying the following algorithm [27]:

(1) $\mathcal{T}^{(0)} = \mathrm{T}$

(2) $\mathcal{T}^{(n)} = \mathrm{T} \,.\, \mathcal{T}^{(n-1)}$

Repeat (2) until $(\mathcal{T}^{(n)} = \mathcal{T}^{(n-1)})$.

Here, $\mathcal{T}^{(i)}$ is the value of $\mathcal{T}$ in iteration i. Also, we borrow the matrix multiplication definition from [27] that states that $C = AB$ is such that:

$C_{ij} = \sum_k(A_{ik}.B_{kj})$

As a result we can perform a per-user inference of trust values for e-mail addresses (associated with users in the OSN) to be added to the user trust repository. In other words, for each user i, the row vector t_i contains trust values for all other users in a maximum distance k from user i. So for every user i we add email addresses associated with t_{ij} in the user trust repository if (i) this information can be shared per user j privacy settings and (ii) $t_{ij} > 0$.

For the other objects (URL, files, IP addresses etc) we adapt the concept of belief from Richardson et al. [27]: any user may assert her personal belief in an object inserted to her trust repository. This personal belief is represented by a value taken from [0,1] (if the user does not assign a value the object is considered trusted, with trust value 1). The higher the value the more trusted is the object. Let b_i represent user i's personal trust value for an object in her trust repository. We refer to the collection of personal beliefs in a particular object as the column vector b. User i can also specify which trusted objects in her trust repository are allowed to be shared. Our next step is to infer how much a user believes in any sharable object in the network. The user trust values (T) will allow us to compute for any user i, her belief in any sharable object using a structure called merged beliefs (b). The merged beliefs structure can be calculated as follows:

(1) $b^{(0)} = \mathrm{b}$

(2) $b^{(n)} = \mathrm{T} \,.\, b^{(n-1)}$, or $(b_i)^n = \sum_k(t_{ik}.(b_k)^{n-1})$

Repeat (2) until $b^{(n)} = b^{(n-1)}$.

Here, $b^{(i)}$ represents the value of b in iteration i. The result of these computations is a merged trust repository with objects and their infered trust values that is sent to the OS via a secure SSL channel.

E. Benefits

Our socially-aware computing paradigm can greatly streamline the design of security policies. Incorporating the role of human users can add diversity to host behavior, decrease the success rate of attacks and thus make a significant impact from the cyber security perspective. Processes, module functions, instructions at the architecture level below a certain threshold of trust level can be prevented from accessing (reading or/and writing or/and executing) a certain range of memory locations, I/O devices or any type of system resource. Raw bytes that fall below a certain threshold of trust value can be prevented from being written into certain memory areas or files.

Anti-SPAM/scam techniques can be improved by associating an e-mail message with a sender trust value. This model also streamlines the decision-making process of installing a new software, downloading a file, or visiting a particular URL if those objects are associated with trust values in a trust repository. Moreover, this model can be extended by allowing the owner of a trust repository to assign certain objects with privacy levels. For example, a browser could use the OS trust interface to allow a user to mark certain information to be filled in a web page as sensitive (e.g. a password or credit card number). This allows the OS to restrict read access to memory locations storing that information, thereby preventing malware from leaking it.

F. Threats to Our Model

What if the OSN infrastructure or the OS kernel is compromised by an attacker that can tamper with the integrity of a trust repository? In this situation an attacker could add malicious objects into a trust repository with a high level of trust. For example, an attacker could assign spamguy@gmail.com or www.maliciousurl.com a high trust value and decrease the trust value of www.amazon.com or any trusted file, IP address or e-mail contact. We argue that for the case of increasing the trust value of malicious objects the system would operate as if a sociallly-aware interface had never existed. Current OSes offer no or little distinction (in terms of trust levels) among objects they access such as files, URLs, email contacts or online friends. For example, if spamguy@gmail.com has a high trust value, all the e-mails from this user would not be flagged as suspicious and all the files downloaded from this e-mail and embedded URLs would have the default trust values in the system. This is exactly what would happen for a system that does not employ our socially-aware model, i.e., that does not make distinctions between trusted and untrusted values. In our model objects and subjects do not have increased privileges in the system because of high trust values. It is actually the other way around: the higher trust values, the closer they are to the default privileges a local file/subject has in the system. These values are never more than the default access control policies provided by the OS, based on the privileges of a user (root or regular user). Our goal here is, instead of considering all objects as trusted or untrusted, provide a continuum of trust

values that allows for stronger but flexible security policies based on user context.

The case where an attacker decreases the values of trusted objects in the system can be described as a form of denial of service attack. For this case, we assume that the OSN provider is fully trusted and the OS kernel has its integrity preserved.

V. Research Challenges

A. Management and Reliability of Social Data and Trust

Our framework emphasizes the importance of adding social information to systems components but social data is heterogenous and highly variable [32] in terms of reliability, for several reasons. First, applications on the Internet from which social data can be derived from have no standardized API. Second, user profiles are not convergent: different sites have unique login credentials. Third, the forms and levels of trust on OSNs vary highly. For example, on Facebook, user trust relationships are mutual. On Twitter, relationships are often unidirectional as follower users choose to follow other users (followed) but the followed user does not have to follow the follower. Given these realities, several research challenges are apparent: (i) extracting reliable information from OSN users, (ii) constraints posed by the ethical issues of mining OSN data, (iii) different challenges for different OSNs, (iv) unpredictability of users in their assignment of trust values, (v) OSNs vary in their conduciveness to assigning trust value to particular objects, and (vi) data in some OSNs are private and in others it is public.

B. The Socially-Aware OS Kernel

Can or should the OS kernel manage more than one trust repository concurrently? The trust repositories may be associated with a particular user in the system and all the processes, files, and objects she owns. However, an important research challenge is how to mirror different trust repositories in a DIFT system at the architecture layer. Other issues are performance, usability, Sybil attacks, and identity management and naming [33].

C. Confidentiality and Security

The trend of online social networking has raised some concerns about new unknown vulnerabilities. For instance, personal information being leaked out implies a violation of user privacy. A research challenge is deriving a probable/specified intention of privacy settings and then determining that appropriate amount/type of social informatics that should be supported on behalf of the user. Social informatics and trust enables a community to share their experience/knowledge to solve large-scale distribution problems collaboratively. For example, user Alice can decide about visiting a URL or installing a new software based on the experience her community (the system administrator at her company or a knowledgeable friend) had with those objects. What should be the right process for the social community to form a converging decision? How should we address potential attacks to this collaborative model? Regarding privacy and anonymity, how can users safely export trust information to their social contacts?

VI. Conclusion

This position paper proposes a novel trustworthy computing paradigm that employs social informatics as a first class resource to manage and leverage social trust to improve security in computing systems. To enhance security and increase cost for attackers this paradigm uniquely includes the role of human users. We leverage social trust because it distinguishes a continuum between totally untrusted and totally trusted values. This continuum can be defined within a socially-aware OS kernel, virtual machine and user applications so that stronger security policies can be deployed.

Acknowledgments

We would like to thank our anonymous reviewers for their comments and Allen Tucker for his suggestions on early drafts of this paper.

References

[1] E. Oswald, "Facebook more popular than google http://technologizer.com/2010/09/10/facebook-more-popular-than-google/."
[2] H. J. Wang, A. Moshchuk, and A. Bush, "Convergence of desktop and web applications on a multi-service os," *USENIX Workshop on Hot Topics in Security*, August 2009.
[3] P. Szor, *The Art of Computer Virus Research and Defense*. 2005.
[4] A. Lanzi, D. Balzarotti, C. Kruegel, M. Christoderescu, and E. Kirda, "Accessminer: Using system-centric models for malware protection," *ACM CCS*, 2010.
[5] S. Forrest, S. A. Hofmeyr, A. Somayaji, and T. A. Longstaff., "A sense of self for unix processes," *IEEE Symposium on S&P*, pp. 120–128, 1996.
[6] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant, "Semantics-aware malware detection," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy (Oakland 2005)*, (Oakland, CA, USA), 2005.
[7] M. Fredrikson, M. Christodorescu, S. Jha, R. Sailer, and X. Yan, "Synthesizing near-optimal malware specifications from suspicious behaviors," *IEEE Symposium on S&P*, 2010.
[8] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," *IEEE S&P*, May 2010.
[9] G. E. Suh, J. Lee, and S. Devadas, "Secure Program Execution via Dynamic Information Flow Tracking," in *Proceedings of ASPLOS-XI*, Oct. 2004.
[10] J. R. Crandall and F. T. Chong, "Minos: Control Data Attack Prevention Orthogonal to Memory Model," *MICRO*, pp. 221–232, December 2004.
[11] J. Newsome and D. Song, "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software," in *NDSS*, Feb. 2005.
[12] G. PortoKalidis, A. Slowinska, and H. Bos, "Argos: an Emulator for Fingerprinting Zero-Day Attacks," *EuroSys*, April 2006.
[13] M. I. Al-Saleh and J. R. Crandall, "On information flow for intrusion detection: What if accurate full-system dynamic information flow tracking was possible?," *New Security Paradigms Workshop*, 2010.
[14] A. Ho, M. Fetterman, C. Clark, A. Warfield, and S. Hand, "Practical taint-based protection using demand emulation," *EuroSys*, 2006.
[15] F. Qin, C. Wang, Z. Li, H. seop Kim, Y. Zhou, and Y. Wu, "LIFT: A Low-Overhead Practical Information Flow Tracking System for Detecting Security Attacks," *MICRO-39*, pp. 135–148, December 2006.
[16] H. Chen, X. Wu, L. Yuan, B. Zang, P. chung Yew, and F. T. Chong, "From Speculation to Security: Practical and Efficient Information Flow Tracking Using Speculative Hardware," *ISCA*, June 2008.
[17] D. Lewis and A. Weigert, "Trust as a social reality," *Social Forces*, vol. 63, no. 4, pp. 967–985, 1985.
[18] N. Luhmann, *Trust and Power*. Wiley, 1979.
[19] D. Gambetta, *Trust: Making and Breaking Cooperative Relations*. Blackwell, 1990.
[20] S. Marsh and P. Briggs, "Examing trust, forgiveness and regret and computational concepts," in *Computing with Social Trust*, pp. 9–42, Springer, 2010.
[21] S. P. Marsh, *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, Apr 1994.

[22] L. Mui, *Computational Models of Trust and Reputation: Agents, Evolutionary Games, and Social Networks*. PhD thesis, Department of Electrical Engineering and Computer Science. Massachusetts Institute of Technology, 2003.
[23] J.-M. Seigner, *Trust, Security and Privacy in Global Computing*. PhD thesis, Trinity College, Dublin, 2005.
[24] J. Golbeck, *Computing and Applying Trust in Web-based Social Networks*. PhD thesis, Department of Computer Science. University of Maryland, College Park, 2005.
[25] A. Abdul-Rahman, *A Framework for Descentralized Trust Reasoning*. PhD thesis, Department of Computer Science. University College London, 2004.
[26] K. Krukow, *Towards a Theory of Trust for the Global Ubiquitous Computer*. PhD thesis, University of Aarhus, 2006.
[27] M. Richardson, R. Agrawal, and P. Domingos, "Trust management for the semantic web," in *Second International Semantic Web Conference*, pp. 351–368, 2003.
[28] R. Guha, R. Kumar, P. Raghavan, and A. Tomkins, "Propagation of trust and distrust," in *World Wide Web (WWW)*, pp. 403–412, 2004.
[29] I. W. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann, 2005.
[30] C. Cortes and V. Vapnik, "Support-vector networks," pp. 273–297, 1995.
[31] D. Oliveira and S. F. Wu, "Protecting Kernel Code and Data with a Virtualization-Aware Collaborative Operating System," *Annual Computer Security Applications Conference (ACSAC)*, December 2009.
[32] J. Golbeck, "Computing with social trust," *Human-Computer Interaction Series*, 2009.
[33] J.-M. Seigner, "Social trust of virtual identities," in *Computing with Social Trust*, pp. 73–118, Springer, 2010.