SECOND INTERNATIONAL WORKSHOP ON AGENTS AND CYBERSECURITY

ACySE2015

Workshop Notes

Natalia Criado
Martin Rehak
Jose M. Such
Laurent Vercouter

May, 5 2015

at AAMAS 2015
Istanbul, Turkey

Workshop Organization

Programme Chairs

Natalia Criado
Martin Rehak
Jose M. Such
Laurent Vercouter

Programme Committee

Javier Bajo
Jim Blythe
Olivier Boissier
Gregory Bonnet
Vicent Botti
Pascal Bouvry
Frances Brazier
Ioana Boureanu
lvaro A. Cardenas
Marco Carvalho
Natalia Criado
Grgoire Danoy
Josep Domingo-Ferrer
Ana Garcia-Fornes
Zahia Guessoum
Chris Hankin
Dieter Hutter
Lucas Kello
Christopher Kiekintveld
Igor Kotenko
Alexei Lisitsa
Ricard Lopez
Emil Lupu
Pasquale Malacaria
Gregorio Martinez
Peter McBurney
Haralambous Mouratidis
Andrea Omicini
Gilbert L. Peterson
Tomas Pevny
Guillaume Piolle
David Pym
Awais Rashid
Omer Rana
Martin Rehak
Michael Rovatsos
Jordi Sabater-Mir
Fabrizio Smeraldi
Jose M. Such
Wamberto Vasconcelos
Laurent Vercouter
Salvatore Vitabile
Martijn Warnier

Prologue

Natalia Criado[1], Martin Rehak[2], Jose M. Such[3], and Laurent Vercouter[4]

[1] Liverpool John Moores University, UK
`n.criado@ljmu.ac.uk`
[2] Cisco Systems, Czech Republic
`marrehak@cisco.com`
[3] Lancaster University , UK
`j.such@lancaster.ac.uk`
[4] Laurent Vercouter (INSA de Rouen/LITIS lab, France)
`laurent.vercouter@insa-rouen.fr`

INTRODUCTION

Research in cyber security is nowadays one of the hottest topics in computer science. This is because security is of capital importance to the development of a sustainable, resilient and prosperous cyber world. This includes protecting crucial assets ranging from Critical Infrastructures to individual's Personal Information, and it spans domains like Cloud computing, Smart grid, Virtual Organisations, Virtual Communities, Social Media, Electronic Commerce and many more. Approaches that are intelligent and self-adaptable are required to deal with the complexities of effectively protecting these crucial assets in all these domains. This is where research from the agent community can make a difference in cyber security. Indeed, cyber security is increasingly receiving more and more attention from the agent community.

The focus of the Second International Workshop on Agents and CyberSecurity (ACySE) is to provide a forum to discuss and advance cyber security by means of agent-based approaches.

Table of Contents

A Vision for Privacy and Transparency through Policy-Carrying Data

Julian Padget
Dept. of Computer Science
University of Bath
Bath, BA2 7AY, U.K.
j.a.padget@bath.ac.uk

Wamberto W. Vasconcelos
Dept. of Computing Science
University of Aberdeen
Aberdeen, AB24 3LT, U.K.
w.w.vasconcelos@abdn.ac.uk

ABSTRACT

As the customer has become the product, so (individual) privacy has become the currency: it is traded for access to resources and paid for in the revelation of many small facts and preferences that are just data on their own, but become information as part of larger population. While fiat currency is traceless and untraceable – it is generally impossible to tell where it has come from or where it has gone – individual data can sometimes be traced back to its origin(ator), but can often be untraceable, once passed on. The concerns expressed at various levels of society are effectively the product of (i) the proliferation of individually attributable (digitally represented) data that can be seen somehow belonging to and having the potential to compromise the individual or be of benefit to a third party and (ii) a regulatory framework that is unprepared for the volume, variety, (or velocity and veracity) of data. In this context, transparency is being proposed as the means to trade off privacy while maintaining security. In response, we propose the notion of data that is inseparable from its access policy and furthermore that any derived data shall also have a suitable derived policy inextricably associated with it. We sketch a framework and some formal notions to capture these ideas to initiate debate.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Policies, data sharing, internet-of-things

Keywords

AAMAS proceedings, LaTeX, text tagging

1. INTRODUCTION

We propose a means to capture the expression of controls over (derived) data and to associate that indivisibly with the data via what we call "policy-carrying data" (PCD[1]). Following policy conventions, in defining the who, the when and the how, the PCD establishes permissions for what the data consumer may do to the data. A novel aspect of our proposal is the establishment of obligations concerning what the consumer should do with the (derived) data and it is these that are the foundation of transparency. Such obligations are the transactional unit for a non-pecuniary data economy, where access to and use of data may be traded for obligations that act as a form of user-definable, liquidity at-point-of-use community currency [6]. These obligations may pertain directly to actions of data consumers or – and this we believe is also a significant novelty of our proposal – indirectly to the policy associated either with the extracted data or the data derived from them.

A feature of the current data landscape is the relative freedom of movement of data from individuals to the data silos used in cloud computing and thence between silos, which could be viewed as contributary to the disempowerment that individuals might feel over their own data – privacy controls aside [1]. The situation is potentially further complicated since the platform may enable the collection and interpretation of those data, thus adding value to them, as in the case of activity-monitoring devices or home energy monitors. The PCD concept associates data with bespoke policies: for example, framework policies might be defined by legislation, while specific policies for individual needs would have to satisfy the norms established at the primary level [5]. In this way, crisp but unworkable definitions of issues such as "When do data stop being private?" and "How to decide if data revelation is in the public interest?" can be blurred as distinctions are established to meet the needs of a given situation.

In the next section we set out a framework for policy-carrying data that considers the information model, identifies classes of stakeholders and puts forward some illustrative examples of natural language expressions of policy (mapping from natural language to formal is a challenging problem for future work). This is followed by a short discussion on the representation of policy and how our perspective draws upon the significant body of work relating to norms and their formalisation, stemming from logic and from multiagent systems. Consequently, we illustrate a possible reasoning framework and its use of state to maintain an audit trail of consumer actions. We conclude with a short discussion of some related work and directions for future work.

2. A FRAMEWORK FOR PCD

We set out a reference framework within which we situate and connect stakeholders, PCDs and an information model. We illustrate our framework in Fig. 1, where we show stakeholders (circles), processes (arrows) and information model (boxes within central box). The stakeholders envisaged are (i) data owners/producers who make data/information available (represented as the left-hand circle) and, in a richer version of the model, these may be separated into those that assert rights over the data and those that publish it; (ii) data consumers who want to access data (represented as the right-hand circle) and again in a richer version of the model, there may be entities that are both consumers and producers of data, either by offering aggregation services or by adding value in some way; (iii) monitors responsible for policing the publication and access activities (upper circle in the middle).

We note that the first two types of stakeholders can be institutions or people as well as computational entities such as sensors,

[1] PCD also stands for "policy-carrying data collection" and we use PCDs (in the plural) to indicate a set of policy-carrying data collections.

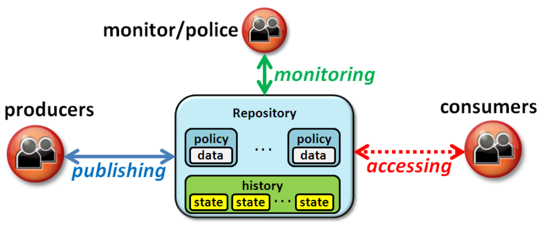


Figure 1: A Framework for Policy-carrying Data

programs, databases, and so on. A monitor works as a third-party authority ensuring that activities (publishing and accessing) follow policies and dealing with violations. Each of these stakeholders has specific ways to interact via the repository: (i) publishing (represented by the blue solid line) is the process whereby data owners/producers make their data available but "wrapped" within a policy, that is, they publish, in a repository, some policy-carrying data. (ii) accessing (represented by the red dotted lines) is the process whereby data consumers *attempt* to obtain access to data mediated via policies. (iii) monitoring (represented by the green line) concerns observing activities and checking for policy compliance or violation, and dispensing rewards or sanctions.

Our framework relies on an information model comprising the PCD – a policy and a data collection made available through the policy – and a history – a collection of events (*i.e.*, a record of activities carried out) gathered at particular time points, denoted as the *states* of the repository. The framework would support stakeholders carrying out the cycle of publish-access-monitor activities supported by a Web server. Servers should be equipped with functionalities to enable the policing of those accessing and uploading PCD, keeping records of usage and (non-)compliance, and enforcing the policies' access control. We envisage programmatic access to PCD, whereby programs and functionalities developed with specific technologies can access any PCD, interacting with the server via pre-establised protocols.

A typical PCD would express something like "Lab managers can access 500 records of my data". If an interested party requested 1,000 records, the server would (i) check the credentials of the requester (who needs to be registered); (ii) grant access to 500 records (a message would provide reasons for not providing the 1,000 records); (iii) update the record of that requester with respect to that PCD. Further requests from the same party would be rejected with a suitable justification. For such control to be in place, the server requires a record of events – an explicit account of the history of the PCD, how they have been used, by whom and when. We observe that PCD can also be used as a means to "wrap" a sensor or other data source, including whole sensor networks. We can have, for instance, a policy establishing that "anyone is permitted to request the temperature reading of the sensor once every hour".

3. POLICY REPRESENTION AND REASONING

Much research has been carried out on data access policies since the early UNIX file systems [9]. Some notable features our approach are as follows. We include means to refer to a history of events; examples are "the first 10 people can use my data" and "anyone is permitted to use n records of my data". We provide fine-grained control over who is to access the data, and under what circumstances; for instance, "invididual i_{285} is forbidden to access my data" and "anyone from company x may use my data after 6PM". Additionally we capture dynamic aspects of data usage, examples being "whoever accesses D_1 should not access D_2" and "anyone who uses my data should provide data".

We adapt and extend current work on normative reasoning for multi-agent systems [2, 3] to represent roles (of participants), data-related events (such as accessing records or publishing data collections), authorship of events and attempts thereof, activation and deactivation conditions of policies, and the object of the policy, namely, the data collection itself. An example policy is:

$$\langle \overbrace{\langle \underbrace{\{\neg accessAll(D_1)\}}_{activation}, \underbrace{\{accessAll(D_1)\}}_{deactivation}, \underbrace{\mathsf{P}_{all}\, accessAll(D_1)}_{target} \rangle}^{policy}, \overbrace{D_1}^{data} \rangle$$

This captures permission to access all records of a data collection. An activation condition says that the permission is in place if the records have not yet been accessed, and the policy is deactivated when the records are accessed, thus providing "one-off" access to the data. The *all* role says anyone can use this policy.

Complementary to the informal discussion of policy examples and and requirements above, we have developed algorithmic specifications of three mechanisms to enable stakeholders to reason with and about their PCDs, so that: (i) a PCD publisher can obtain the identity of individual agents who have access (via their associated roles) to data collections. (ii) an agent a can analyse a set of PCDs and gather all the obligations which might apply to it (iii) access can be policed, through a language comprising permissions and prohibitions (that are use as a means for temporary derogation of permissions), and also logged. The formalization of the model and the details of the above algorithms appear in [7]. Meanwhile a detailed comparison with [4] and [8] is in preparation.

REFERENCES

[1] L. Brandimarte, A. Acquisti, and G. Loewenstein. Misplaced confidences: Privacy and the control paradox. *Social Psychological and Personality Science*, 4(3):340–347, 2013.

[2] M. Şensoy, T. J. Norman, W. W. Vasconcelos, and K. Sycara. OWL-POLAR: A framework for semantic policy representation and reasoning. *Web Semantics: Science, Services and Agents on the World Wide Web*, 12-13, 2012.

[3] A. García-Camino, J. A. Rodríguez-Aguilar, C. Sierra, and W. W. Vasconcelos. Constraint rule-based programming of norms for electronic institutions. *Autonomous Agents and Multi-Agent Systems*, 18(1):186–217, 2009.

[4] G. Karjoth, M. Schunter, and M. Waidner. Platform for enterprise privacy practices: Privacy-enabled management of customer data. In R. Dingledine and P. F. Syverson, editors, *Privacy Enhancing Technologies, PET 2002, Revised Papers*, volume 2482 of *LNCS*, pages 69–84. Springer, 2002.

[5] T. Li, T. Balke, M. De Vos, J. A. Padget, and K. Satoh. A model-based approach to the automatic revision of secondary legislation. In *International Conference on Artificial Intelligence and Law*. ACM, 2013.

[6] B. Litaer. *The Future of Money: Creating New Wealth, Work and a Wiser World*. Century, 2002.

[7] J. Padget and W. Vasconcelos. Policy-carrying data: A step towards transparent data sharing. In *Proceedings of 6th International Conference on Ambient Systems, Networks and Technologies, ANT 2015*. Elsevier, June 2015. In press.

[8] S. Pearson and M. Casassa Mont. Sticky policies: An approach for managing privacy across multiple parties. *IEEE Computer*, 44(9):60–68, 2011.

[9] V. Suhendra. A survey on access control deployment. In *Security Technol.*, volume 259 of *Comm. in Comp. & Inf. Science*. Springer, 2011.

Argumentation for Multi-party Privacy Management

(Position Paper)

Ricard Fogues
Universitat Politècnica de València
Camí de Vera, SN
Valencia, Spain
rilopez@dsic.upv.es

Pradeep Murukannaiah
North Carolina State University
Raleigh, NC, USA
pmuruka@ncsu.edu

Jose M. Such
Lancaster University
Lancaster, UK
j.such@lancaster.ac.uk

Agustin Espinosa
Universitat Politècnica de València
Camí de Vera, SN
Valencia, Spain
aespinos@dsic.upv.es

Ana Garcia-Fornes
Universitat Politècnica de València
Camí de Vera, SN
Valencia, Spain
agarcia@dsic.upv.es

Munindar Singh
North Carolina State University
Raleigh, NC, USA
mpsingh@ncsu.edu

ABSTRACT

Social network services enable users to share large quantities of private information. Often, the shared information concerns individuals who are members of the social network but did not upload the information to the service. In such situations, inappropriate sharing preferences can cause conflict and threaten users' privacy. Since related studies suggest that users prefer to solve multi-party privacy conflicts through negotiation, we introduce a novel approach based on negotiation through arguments. In our approach, users propose privacy settings and support their proposals with logical arguments. The final decision is based on a setting supported by sound arguments.

Categories and Subject Descriptors

K.4.1 [**Computers and Society**]: Privacy; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

Keywords

Privacy, Social Network, Argumentation

1. INTRODUCTION

A social network service (SNS) enables users to maintain social relationships via online interactions. As users on an SNS interact, they share information with each other. Often, the information shared on an SNS involves several users (e.g., a photo showing a group of people). Many SNSs enable their users to connect the information they upload to other users; these connections are usually employed to notify the concerned users that the information was uploaded. Since the information shared varies depending on the SNS, (e.g., Instagram is focused only on photos and Twitter on short textual messages), these connections can take different forms, e.g., tags on a photo or mentions in a tweet. For example, Alice uploads a photo from last weekend's party where she and her friend Bob appear together, and tags Bob on the picture. When these connections are created, the other users are also linked to the uploaded information. Usually, a connection implies that the profile of the user can be accessed from the information or some personal information is shown in conjunction with the uploaded data. Although connections between information and users are widely employed by SNS users, they can also pose a privacy threat. For example, Bob thinks that the photo uploaded by Alice is somewhat sensitive and he is not sure about uploading it to an SNS. However, since Bob has no control over uploading that photo, Alice's action can threat Bob's privacy. We identify situations like this as multi-party privacy conflicts.

Currently, SNSs do not have mechanisms to handle multi-party privacy conflicts [**?**]. Thus, a user who did not upload a piece of information concerning him has to either agree with the sharing preferences chosen by the uploader or remove the connection that links the user to the shared information. A trivial approach to solve such conflicts is to respect the sharing preferences of every party. However, the nature of conflicts in preferences can make this solution inviable. For example, according to their individual privacy preferences, Alice would like to share the photo with Charlie but Bob would like to share it only with common friends and he does not know Charlie. Here, there is no solution that completely respects both parties' preferences.

Decision support systems that help users resolve multi-party privacy conflicts have been identified as one of the biggest gaps in privacy management in social media [2, 10, 19, 14]. The main challenge for these systems is to propose solutions that can be accepted most of the time by all the users involved, minimising the burden on the users to resolve multi-party privacy conflicts.

Based on evidence that users negotiate over what privacy settings they should employ [10, 19], we hypothesize that, during the negotiation of setting a privacy preference, users employ arguments to convince the other parties that their demands are reasonable and should be taken into account. Therefore, we propose a new approach for managing multi-party privacy conflicts based on logical arguments. The forms of these arguments can be classified in a number of schemes, such as precedence or popular opinion. Besides, other variables, such as relationship types among the users, can play a key role during the negotiation and in the final decision.

2. RELATED WORK

One may think that the most direct approach to manage multi-party privacy policies is employing veto voting, as already suggested in [16]. That is, denying access takes precedence over granting access. Thus, if an individual wants to share the information with a given user, but another individual does not, the information is not shared. The obvious benefit of this approach is that it does not allow privacy breaches. However, there is a problem that advises against always employing this solution. Since denying access takes precedence, there may be cases in which veto voting leads to sharing utility loss. For example, Alice and Bob appear together in a photo. Bob initially opposes sharing the photo with Charlie as he does not know him. However, if Alice tells him that Charlie is her friend and that everything is ok, then Bob may accept sharing with Charlie. Had the veto voting been applied, then the item would have not been shared with Charlie, being a missed opportunity to share.

There are other proposals in the related literature that aim to help users resolve multi-party privacy conflicts [18, 11, 3, 8, 7]. However, some of them [18, 11] need too much human intervention during the conflict resolution process to be practicable, by requiring users to solve the conflicts *manually* [18] or very close to *manually* [11], e.g., participating in difficult-to-comprehend auctions with fake money for each and every possible conflict. Other approaches to resolve multi-party privacy conflicts are more automated [16, 3, 8], but they only consider one fixed way of aggregating user's privacy preferences without considering how users would actually achieve compromise and the concessions they might be willing to make to achieve it depending on the specific situation. Only [7] considers more than one way of aggregating users' privacy preferences, but the user that uploads the item chooses the aggregation method to be applied, which becomes a unilateral decision without considering any input from others. Clearly, solutions that do not consider input from all the users involved may lead to solutions that are far from what some users would be willing to accept. All of this causes these more automated mechanisms to have difficulties to adapt to different situations that may motivate different users' concessions, which has the potential to cause these mechanisms to suggest solutions that may not be acceptable by all users, so that users may need to end up resolving multi-party conflicts manually most of the time. The work presented in [13] provides an improvement over these fixed ways of aggregating privacy preferences from users by suggesting 3 different methods that would be selected depending on the particular situation, but again, only a limited number of aggregation methods is considered.

Finally, some very recent works propose game-theoretic negotiation mechanisms to tackle the multi-party privacy conflict resolution problem [9, 15]. These proposals provide an elegant analytic framework to study the problem and the kind of solutions that can be obtained based on well-known solution concepts such as the Nash equilibrium. However, as shown in [9], these proposals may not work well in practice since it seems they may not be able to capture well the social idiosyncrasies users actually consider in the real life when they face multi-party privacy conflicts [10, 19].

3. PROBLEM STATEMENT

In this paper, we propose a novel approach to resolve multi-party privacy conflicts based on argumentation. As in [2, 18], users negotiate over what sharing preferences should be applied. However, in our approach, users do not classify their preferences by their strength. Instead, their preferences have to be supported by logical arguments. As in a real discussion, the sound arguments are the ones considered in the conclusion of the negotiation, while weak or unsound arguments are discarded. This means that the preferences supported by the sound arguments will be applied, or at least, taken into consideration.

We model the multi-party privacy management problem as a multi-agent scenario. Each individual linked to a piece of information to be shared on an SNS employs a personal agent that acts on his or her behalf during the negotiation. These personal agents consider the following variables during the negotiation:

Ownership : An individual can play one of two roles, the owner of the information, or a stakeholder—an individual who is somehow linked to the data but is not the owner. As concluded by Besmer et al. [2], SNS users believe that being the owner of the information implies some authority over the final sharing settings. Thus, during the negotiation, agents must be aware of this position of power of the owner and behave accordingly.

Relationships among the individuals : An individual has specific types of social relationship with other individuals, e.g., close friend or sibling. The closeness and authority relationships can modify how a person negotiates. Closeness can affect on how much an opinion is taken into account. For example, it is very likely that a user's sibling respects the user's preferences. On the other hand, a person with authority over others can impose her opinion even when all the others have different views.

Sensitivity of the information : The appropriateness to share something on an SNS is subjective. For example, in some cultures drinking alcohol is a taboo, thus, a photo showing a person drinking can be inappropriate, where as it can normal in other cultures. Each individual involved in the negotiation has a perception of the sensitivity of the information. This perception will affect how that person tries to impose her view.

The goal of each agent during the negotiation is to apply a sharing preference to the information as close as possible to its principal's preference. The preferences of two agents can be compatible or conflicting with each other. For example, the preference of Bob: "I do not want my parents to see this photo" is compatible with the preference of Alice: "I want only my friends to see it", as long as Alice's friends do not include Bob's parents.

The negotiation consists of a predetermined number of rounds. During each of these rounds the agents propose privacy settings supported by arguments. An agent's arguments and settings can change and adapt to what other agents propose from round to round. After the preset number of rounds of negotiation, a final decision is made. This decision should respect the preferences of every individual involved to the best possible extent. If all preferences are compatible with each other, the solution is trivial. However, in case of conflicts, the preferences supported by the sound arguments should take precedence over the others. Since creating such sharing preferences can be difficult for an average SNS user, this process should be automated, so that a sharing preference recommender suggests the final sharing configuration.

Consider an example negotiation between Alice and Bob:

1. Bob: It's a funny photo, but embarrassing since I appear drunk. I don't want strangers seeing it.
2. Alice: We had a lot of fun during the party. Everybody's talking about how funny you were and they want to see your photos. Let's share it with everybody.
3. Bob: Well, if you insist... we should share it like we always share photos like this, only with common friends.
4. Alice: C'mon! it was our graduation party! It's something that we only do once in our lifetime. We should show it to the world.

In this example, Alice is the owner. She and Bob are close friends, thus, they negotiate using informal language. Alice thinks that the photo is not very sensitive, while Bob thinks the contrary. In this example, there are two rounds of negotiation and in each round, Alice and Bob give arguments to support why they should share the photo in a specific way.

4. ARGUMENTS

In our approach, agents employ arguments to convince the other parties that their privacy preferences should be used, or at least considered for the final decision. Different types of arguments can be employed during a negotiation. Each of these arguments corresponds to an argumentation scheme [17]. These schemes are argument forms that represent inferential structures of arguments used in everyday discourse. Although, given the appropriate situation, almost every argumentation scheme can be used during a negotiation, we hypothesize that in our context only four schemes fit: argument (i) from consequences, (ii) from analogy, (iii) for an exceptional case, and (iv) from popular opinion. Argumentation schemes work as classes of arguments and each argument employed by an agent is an instance of a class.

We consider that individuals value their relationship with others, thus, they are well intended. Consequently, individuals do not employ fallacies during the negotiation and the arguments used are based on actual facts. The goal of an agent is to guarantee that its privacy preferences are respected as much as possible. Thus, the agents are likely to employ the arguments that are the most convincing.

Argument from Consequences: If A is brought about, then good (bad) consequences will occur. Therefore, A should (not) be brought about. An example of good consequences in our context is: *We had a lot of fun during the party. Everybody's talking about how funny you were and they want to see your photos. Let's share it with everybody.* An example of argument from bad consequences is: *It's a funny photo, but embarrasing since I appear drunk. I don't want strangers seeing it.*

When an SNS user shares something, she expects to obtain some kind of benefit in terms of friendship, jobs, and other social opportunities [5]. Therefore, it is reasonable to argue that sharing certain information implies a good consequence. On the other hand, sharing inappropriate information can harm people's feelings and cause social tensions. Thus, negative consequences can also form a valid argument.

Argument from Analogy: Generally, case C_1 is similar to case C_2 and A is true (false) in case C_1. Then, A is true (false) in C_2. An example of an argument using this scheme is: *We should share it like we always share photos like this, only with common friends.*

We find a number of approaches for managing privacy on SNS based on tools that automatically suggest privacy preferences [1, 4, 6, 12]. These tools employ past privacy settings employed by the user to infer new configurations. Many of these tools have been developed and evaluated satisfactorily with users. Thus, past decisions can be exploited for suggesting new privacy settings.

Argument for an Exceptional Case: If the case of x is an exception, then the established rule can be waived in the case of x. An example in our domain is: *C'mon! it was our graduation party! It's something that we only do once in our lifetime. We should show it to the world.*

Although previous privacy configurations can act as a guide for future elements, exceptions need a different approach. The scheme for an exceptional case is, at some level, the opposite to the scheme of argument from analogy. The arguments created from this scheme cover cases where privacy recommending tools would fail. Obviously, an individual has to make a hard case to justify why the new element is such an exception that needs a different consideration to the other previous and similar elements.

Argument from Popular Opinion: If the large majority in a particular reference group G accepts A as true (false), then there exists a presumption in favor of (against) A. An example is: the majority of the people that appear in the photo think that it should be kept private. Therefore, we should not share it with anyone.

Argument created from this scheme can take two forms: (i) they can be explicitly employed in an utterance, or (ii) they can emerge from the suggested privacy settings supported by other arguments. The first form can only be used in the following round of negotiation. It is not possible to use it in the first round as individuals still do not know what others' opinions are. The second form for an argument from popular opinion automatically emerges when two or more individuals suggest the same sharing preferences. Thus, although no individual explicitly employed an argument from popular opinion, it is considered for the final outcome.

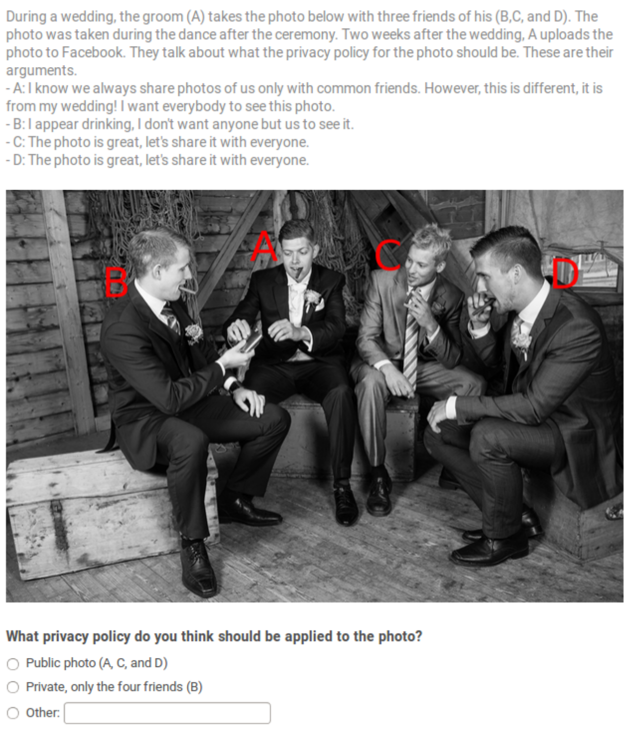
5. OPEN CHALLENGES

The goal of our research is to build a privacy recommender tool that helps users to decide what sharing configuration they should apply to a piece of information that concerns several individuals. This goal entails a number of challenges that future work should address.

First, since the recommender must provide suggestions that are similar to what humans do, we need to collect data from human participants. However, generating real scenarios where privacy conflicts arise is nontrivial. Hence, we plan to survey SNS users using hypothetical situations that present such conflicts. The variety of situations must be sufficient to collect several instances of every possible combination of variable values and arguments. Figure 1 shows an example of a possible hypothetical scenario presented to the participants.

The data collected will be used to generate a predictive model that, given a set of arguments, privacy settings, relationship types, sensitivity values, and roles as input, provides a privacy setting as output. The predictive model can be based on a machine learning technique (e.g., decision trees) or heuristics. The model will be trained and tested with the data collected from the real SNS users.

Besides the variables proposed, it is worth noting that social pressure might play a role. That is, if the majority of


During a wedding, the groom (A) takes the photo below with three friends of his (B,C, and D). The photo was taken during the dance after the ceremony. Two weeks after the wedding, A uploads the photo to Facebook. They talk about what the privacy policy for the photo should be. These are their arguments.
- A: I know we always share photos of us only with common friends. However, this is different, it is from my wedding! I want everybody to see this photo.
- B: I appear drinking, I don't want anyone but us to see it.
- C: The photo is great, let's share it with everyone.
- D: The photo is great, let's share it with everyone.

What privacy policy do you think should be applied to the photo?

○ Public photo (A, C, and D)
○ Private, only the four friends (B)
○ Other:

Figure 1: Example of hypothetical scenario.

users involved advocate for a privacy policy, this might have an effect on the final decision. Indeed, most of current approaches to this problem apply some kind of voting mechanism: majority voting, veto voting, and uploader overwrites. To evaluate the effect of social pressure, it is necessary to create scenarios where a majority of users want something and there is a user that opposes strongly.

In our approach, we assume that agents do not employ fallacies and that arguments are always valid. This assumption reduces the complexity. However, future work should look into ensuring the validity of arguments and punishing or applying penalties to the agents that use invalid arguments.

REFERENCES

[1] S. Amershi, J. Fogarty, and D. Weld. Regroup: Interactive machine learning for on-demand group creation in social networks. In *Proc. of CHI 30th*. ACM, 2012.

[2] A. Besmer and H. Richter Lipford. Moving beyond untagging: photo privacy in a tagged world. In *Proc. of the CHI 28th*, pages 1563–1572. ACM, 2010.

[3] B. Carminati and E. Ferrari. Collaborative access control in on-line social networks. In *IEEE CollaborateCom*, pages 231–240, 2011.

[4] G. P. Cheek and M. Shehab. Policy-by-example for online social networks. In *SACMAT '12*, pages 23–32, New York, NY, USA, 2012. ACM.

[5] N. Ellison, C. Steinfield, and C. Lampe. The benefits of facebook friends: Social capital and college students' use of online social network sites. *Computer-Mediated Communication*, 12(4), 2007.

[6] L. Fang and K. LeFevre. Privacy wizards for social networking sites. In *Proc. of WWW 19th*, pages 351–360. ACM, 2010.

[7] R. Fogues, J. M. Such, A. Espinosa, and A. Garcia-Fornes. Open challenges in relationship-based privacy mechanisms for social network services. *International Journal of Human-Computer Interaction*, (just-accepted), 2015.

[8] H. Hu, G. Ahn, and J. Jorgensen. Multiparty access control for online social networks: model and mechanisms. *IEEE TKDE*, 2013.

[9] H. Hu, G.-J. Ahn, and J. Jorgensen. Detecting and resolving privacy conflicts for collaborative data sharing in online social networks. In *Proc. ACSAC*, pages 103–112. ACM, 2011.

[10] H. Hu, G.-J. Ahn, Z. Zhao, and D. Yang. Game theoretic analysis of multiparty access control in online social networks. In *Proceedings of SACMAT '14*, pages 93–102, New York, NY, USA, 2014. ACM.

[11] A. Lampinen, V. Lehtinen, A. Lehmuskallio, and S. Tamminen. We're in it together: interpersonal management of disclosure in social network services. In *Proc. CHI*, pages 3217–3226. ACM, 2011.

[12] A. Squicciarini, M. Shehab, and F. Paci. Collective privacy management in social networks. In *Proceedings of the WWW 18th*, pages 521–530. ACM, 2009.

[13] A. Squicciarini, S. Sundareswaran, D. Lin, and J. Wede. A3p: adaptive policy prediction for shared images over popular content sharing sites. In *Proc. of Hypertext 22nd*, pages 261–270. ACM, 2011.

[14] J. M. Such and N. Criado. Adaptive conflict resolution mechanism for multi-party privacy management in social media. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 69–72. ACM, 2014.

[15] J. M. Such, A. Espinosa, and A. García-Fornes. A survey of privacy in multi-agent systems. *The Knowledge Engineering Review*, 29(03):314–344, 2014.

[16] J. M. Such and M. Rovatsos. Privacy policy negotiation in social media. *arXiv preprint arXiv:1412.5278*, 2014.

[17] K. Thomas, C. Grier, and D. Nicol. unfriendly: Multi-party privacy risks in social networks. In *Privacy Enhancing Technologies*, volume 6205 of *Lecture Notes in Computer Science*, pages 236–252. Springer, 2010.

[18] D. Walton, C. Reed, and F. Macagno. *Argumentation Schemes*. Cambridge University Press, 2008. Cambridge Books Online.

[19] R. Wishart, D. Corapi, S. Marinovic, and M. Sloman. Collaborative privacy policy authoring in a social networking context. In *Proc. of POLICY '10*, 2010.

[20] P. Wisniewski, H. Lipford, and D. Wilson. Fighting for my space: Coping mechanisms for sns boundary regulation. In *Proc. CHI*, pages 609–618. ACM, 2012.

Optimal Network Security Hardening Using Attack Graph Games

Karel Durkota
Dept. of Computer Science
FEE, CTU in Prague
durkota@agents.fel.cvut.cz

Viliam Lisý
Dept. of Computer Science
FEE, CTU in Prague
lisy@agents.fel.cvut.cz

Christopher Kiekintveld
Dept. of Computer Science
Univ. of Texas at El Paso, USA
cdkiekintveld@utep.edu

Branislav Bošanský
Dept. of Computer Science
Aarhus University, Denmark
bosansky@cs.au.dk

ABSTRACT

Preventing the attacks in a computer network is the core problem in network security. We introduce a new game-theoretic model of the interaction between a network administrator who uses limited resource to harden a network and an attacker who follows a multi-stage plan to attack the network. The possible plans of the attacker are compactly represented using attack graphs, while the defender adds fake targets (honeypots) to the network to deceive the attacker. The compact representation of the attacker's strategies presents a computational challenge and finding the best response of the attacker is NP-hard. We present a solution method that first translates an attack graph into a MDP and solves it using policy search with a set of pruning techniques. We present an empirical evaluation of the model and solution algorithms, evaluating scalability, the types of solutions that are generated for realistic cases, and sensitivity analysis.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Economics, Performance, Experimentation

Keywords

Attack graphs, Optimal attack policy, Markov decision process, And-or graph, Honeypots, Game theory, Network security

1. INTRODUCTION

Networked computer systems support a wide range of critical functions in both civilian and military domains. Securing this infrastructure is extremely costly and there is a need for new automated decision support systems that aid human network administrators to detect and prevent attacks.

We focus on network security hardening problems in which a network administrator (defender) reduces the risk of attacks on the network by introducing honeypots (fake hosts or services) into their network [27]. Legitimate users do not interact with honeypots; hence, honeypots act as decoys and distract attackers from the real hosts, send intrusion detection alarms to the administrator, and/or gather detailed information the attacker's activity [26, 10]. However, believable honeypots are costly to set up and maintain. Deciding how to optimally allocate these resources to reduce the risk of attacks on a network presents a challenging decision for the defender. On the other hand, a well-informed attacker should anticipate the use of honeypots and try to avoid them.

We use *game theory* to model this adversarial interaction and to determine the best way to use honeypots against a well-informed attacker. We introduce a novel game-theoretic model of network hardening using honeypots that extends the existing line of Stackelberg security games [32] by combining two elements: (1) we adopt a compact representation of strategies for attacking computer networks called *attack graphs*, (2) the defender uses *deception* instead of directly allocating resources to harden targets.

Attack graphs (AGs) can represent a rich space of sequential attacker actions for compromising a specific computer network. They can be automatically generated based on known vulnerability databases [14, 24] and they are widely used in the network security to calculate security risk measures (e.g., the probability of a successful attack) [21, 13] or identify the minimal subset of vulnerabilities/sensors to be fixed/placed to prevent all known attacks [31, 20]. However, these approaches do not capture the attacker's possible counter-strategies to the administrator's actions and can easily bypass less expensive hardening actions which could suffice to deter the attacker from attacking the network. Attack graphs, beside the researchers' interest, found to be practical in the commercial sector as well, which is evidenced by the fact that cyber-security analytical companies, e.g., Skybox Security [1] or Core Security Technologies [2], use them as described in their patents in [7] and [22]. We use AGs as a compact representation of an attack plan library, from which the rational attacker chooses the optimal contingency plan to follow. Contingency plan prescribes an action to every possible situation that can occur during the attack. Although computing the optimal contingency attack plan in an attack graph is an NP-hard problem [9], the information it provides about the attacker's motivations and likely action order is crucial from the security hardening perspective. We address the complexity issue by translating attack graphs into an MDP and introducing a collection of pruning techniques that considerably reduce the computation. Fast alternative is to generate linear plans that may be provided by classical planners (e.g., [23, 4]), however, they are not sufficient as they cannot represent attacker's behavior after action failures.

Although in this paper we assume rational attackers with known payoffs, the framework of game theory allows extending the model to a whole variety of attackers, such as attacker's with *bounded*

[1]http://www.skyboxsecurity.com
[2]http://www.coresecurity.com

rationality [29], or to assume more attacker types (i.e., *Bayesian games* [33]), or limiting the knowledge learnt about the network (*imperfect information game* [28]), etc.

Deploying honeypots changes the structure of the network and increases uncertainty for the attacker. In the presented game model we assume that the attacker knows the number of deployed honeypots and their type (e.g., a database server). However, the attacker does not know which specific hosts are honeypots and which are real. While the assumption that the attacker knows the number/type of honeypots is strong, it corresponds to a worse-case, well-informed attacker.

We present five main contributions: (1) a novel game-theoretic model of security hardening based on attack graphs, (2) algorithms for analyzing these games, including fast methods based on MDPs for solving the attacker's planning problem, (3) a case study analyzing the hardening solutions for sample networks, (4) empirical evaluation of the computational scalability and limitations of the algorithms, and (5) sensitivity analysis for the parameters used to specify the model.

2. NETWORK HARDENING GAME USING HONEYPOTS

In this section we introduce a game-theoretic model for the network hardening problem. Our model is a Stackelberg game, where the defender acts first, taking actions to harden the network by adding honeypots (HPs). The attacker is the follower that selects an optimal attack plan based on (limited) knowledge about the defender's strategy. In particular, we assume that the attacker learns the number and type of HPs added to the network, but *not* which specific hosts are real and fake.

An instance of the game is based on a specific computer network like the one shown in Fig. 1c (based on [12]). A network has a set of host types T, such as firewalls, workstations, etc. Two hosts are of the same type if they run the same services and have the same connectivity in the network (i.e., a collection of identical workstations is modeled as a single type). All hosts of the same type present the same attack opportunities, so they can be represented only once in an attack graph without omitting any attack options. During an attack, a specific host of a given type is selected randomly with uniform probability and the selection remains the same until the end of the game.

More formally, a computer network $y \in \mathbb{N}^T$ contains y_t hosts of type $t \in T$. The defender can place up to k honeypots into the network y, so his actions are represented by $x \in X \subset \mathbb{N}_0^T$ with $\sum_{t \in T} x_t \leq k$, specifying that x_t hosts type $t \in T$ will be added to the network as honeypots (e.g., by duplicating the configurations of the real hosts with obfuscated data). The modified network consists of $z_t = x_t + y_t$ hosts of type t. Adding more HPs of a specific type increases the likelihood that the attacker who interacts with this type of host will choose a HP instead of a real host. If the attacker interacts with a HP during an attack, he is immediately detected and the attack ends. The attacker is rational and maximizes the expected utility taking into account the probabilities of interacting with HPs, his actions' costs and success probabilities, and rewards from successful attacks. He selects his attack strategy from set Ξ defined later. Installing and maintaining HPs has a cost for the defender depending on the host type ($c(t)$ $t \in T$) that is duplicated. The defender minimizes his total expected loss l which consists of (1) the expected loss for the attacked hosts and (2) the cost for adding the HPs into the network. The Stackelberg equilibrium is found by selecting the pure action of the defender that minimizes the expected loss under the assumption that the attacker will respond with an optimal attack [8]. If the attacker is indifferent between multiple attacks, it is typical to break ties in favor of the defender [32]. The defender's action is

$$x^* = \operatorname*{argmin}_{x \in X} \{ l(x, \operatorname*{argmax}_{\xi \in \Xi} \{ \mathbb{E}(\xi, x) \}) \}. \tag{1}$$

The minimization over all defender actions is performed by systematically checking each option; however, the main computation burden of computing the optimal attack strategy is substantially reduced by caching and reusing results of subproblems that are often the same. Computation of this equilibrium relies on computing the optimal attack policy as explained in the following section.

3. ATTACK GRAPHS

There are multiple representations of attack graphs common in the literature. We use *dependency attack graphs*, which are more compact and allow more efficient analysis than the alternatives [23]. Fig. 1a is an example attack graph with high-level actions for illustration. Formally, it is a directed AND/OR graph consisting of fact nodes F (OR) and action nodes A (AND). Every action has *preconditions* ($\text{pre}(a) \subseteq F$) – a set of facts that must be true before the action can be performed – and *effects* ($\text{eff}(a) \subseteq F$) – a set of facts that become true if the action is successfully performed. These relations are represented by edges in the attack graph. We use the common monotonicity assumption [1, 24, 19] that once a fact becomes true during an attack, it can never become false again as an effect of any action.

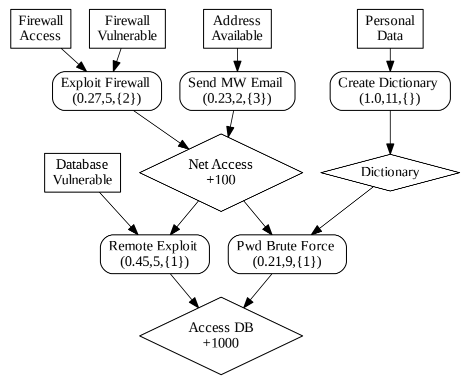
Every action has associated a probability of being performed successfully $p_a \in (0,1]$, and cost $c_a \in \mathbb{R}^+$ that the attacker pays regardless of whether the action is successful. The costs represent the time and resources for the attacker to perform the action. Finally, every action a interacts with a set of host types $\tau_a \subseteq T$. The first time the attacker interacts with a type t, a specific host of that type is selected with uniform probability. Future actions with the same host type interact with the same host. There is no reason to interact with a different host of the same type because (1) rewards are defined based on the types, so there is no additional benefit, and (2) interacting with another host increases the probability of interacting with a honeypot and ending the game. Each fact $f \in F$ has associated an immediate reward $r_f \geq 0$ that the attacker receives when the fact becomes true (e.g., an attacker gains reward by gaining access to a particular host or compromising a specific service). At any time we allow the attacker to terminate his attack by a termination action.

An illustrative example of attack graph is depicted in Fig. 1a. Diamonds and rectangles are fact nodes that are initially *false* and *true*. Actions (rounded rectangles) are denoted with a label and a triple (p_a, c_a, τ_a). The attack proceeds in a top-down manner. At the beginning the attacker can perform actions *Exploit-Firewall*, *Send-MW-Email* or *Create-Dictionary*. The action *Exploit-Firewall*'s preconditions are {*Firewall-Access*, *Firewall-Vulnerable*} and its effect is {*Net-Access*}. If this action is performed, the attacker immediately pays cost $c_a = 5$, interacts with host types $\tau_a = \{1\}$, and with probability $p_a = 0.27$ the action's effects become true. In that case the attacker obtains reward +100.

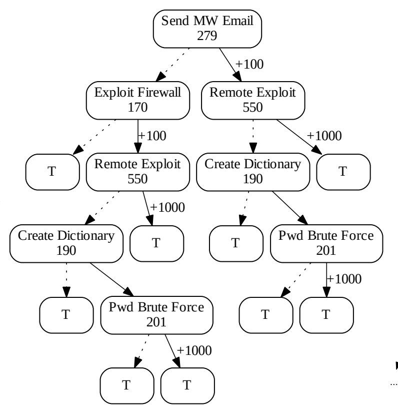
Attack graphs can be automatically generated by various tools. We use the MulVAL [25], which constructs attack graphs from information automatically collected by network scanning tools, such as Nessus[3] or OpenVAS[4]. Previous works (e.g., [30]) show that

[3]http://www.nessus.org

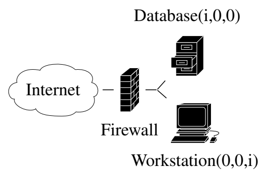
[4]http://www.openvas.org


Firewall Access
Firewall Vulnerable
Address Available
Personal Data
Exploit Firewall (0.27,5,{2})
Send MW Email (0.23,2,{3})
Create Dictionary (1.0,11,{})
Database Vulnerable
Net Access +100
Dictionary
Remote Exploit (0.45,5,{1})
Pwd Brute Force (0.21,9,{1})
Access DB +1000

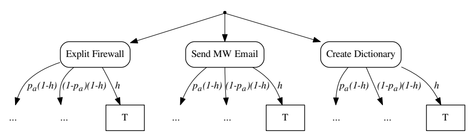
(a) Attack graph



(b) Attack policy



(c) Local-*i* network



(d) Decision point

Figure 1: (a) Attack graph representing the possible ways of gaining an access to the database. (b) Optimal attack policy for the attack graph in (a) where T denotes termination of the attack; (c,d,e) network topologies with triples (r,l,c) at hosts denoting number of remote (r), local (l) and client (c) vulnerabilities at that host.

the information about the costs and success probabilities for different actions can be estimated using the Common Vulnerability Scoring System [18] values available in the National Vulnerability Database [2], historical data, or be directly specified by the network administrator. Although these tools construct attack graphs based on all *known* vulnerabilities, they can represent a great majority of the attacks against which we protect a network.

3.1 Optimal Attacker Policy

In order to fully characterize the attacker's reaction to the set of honeypots, we need to compute a full *contingent attack policy*, which defines an action for each situation that may arise during an attack. This allows identifying not only the actions likely to be executed by a rational attacker, but also the *order* of their execution. This is necessary to evaluate effectiveness of placing honeypots or any other intrusion detection sensors.

The attack strategies Ξ are all contingent plans consistent with the attack graph. The optimal attack policy maximizes attacker's expected utility and in case of ties favors the defender. Fig. 1b depicts the optimal attack policy for the attack graph in Fig. 1a without any honeypots. Nodes represent suggested actions to perform with their expected rewards if strategy is followed. The first action suggested by this policy is to *Send-MW-Email*. If the action is successful, the attacker immediately obtains reward +100 for reaching *Net-Access* and follows the right (sub)policy (solid arc). If *Send-MW-Email* fails, attacker's best choice is to perform *Exploit-Firewall* and expect reward 170. The attack terminates (T) if there are no actions to perform or the expected rewards do not surpass the expected cost of the actions.

4. SOLVING AG USING MDP

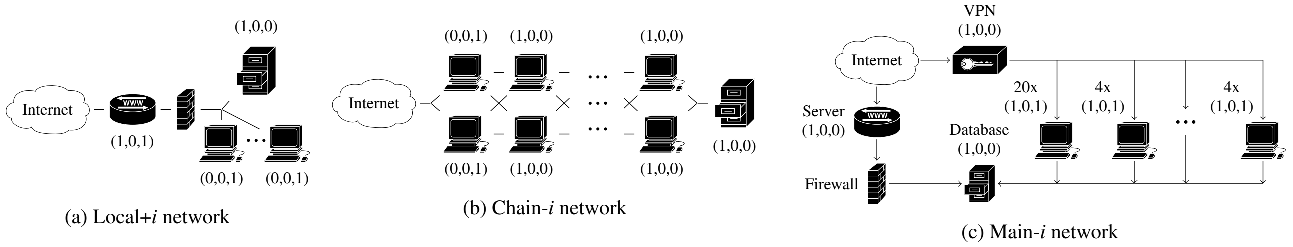
In the Stackelberg game model we assume that the attacker observes the defender's strategy, which can be used in computing the best-response attack strategy. We represent the problem of computing the attacker's best response as a finite horizon Markov Decision Process (MDP) [3]. The MDP for attack graph *AG* is defined as a tuple $\langle B,S,P,\rho\rangle$, where: (1) $B = A \cup \{\mathrm{T}\}$ is the set of actions, (2) S is set of states $s = (\alpha_s, \phi_s, \tau_s)$, where: $\alpha_s \subseteq B$ is the set of actions that a rational attacker can still perform, $\phi_s \subseteq F$ is the set of achieved facts, and $\tau_s \subseteq T$ is the set of host types that the attacker has interacted with so far. The initial MDP state is $s_0 = (B, \emptyset, \emptyset)$. (3) $P^a_{ss'} \in (0,1]$ is the probability that performing action a in state s leads to state s'. Performing action a can lead to one of three possible outcomes: either (i) a interacts with a honeypot with probability $h = 1 - \prod_{t:\tau_a\setminus\tau}(\frac{z_t - x_t}{z_t})$ and his attack immediately ends; (ii) action a does not interact with a honeypot and is successful with probability $p_a(1-h)$ or (iii) action a does not interact with a honeypot and neither is successful with probability $(1-p_a)(1-h)$. The sets defining the newly reached state are updated based on these outcomes. Finally, (4) $\rho^a_{ss'}$ is the attacker's reward for performing action a in state s leading to s'. It is based on the set of facts that became true and the action cost c_a.

We compute the optimal policy in this MDP using backward induction based on depth-first search, with several enhancements to speed up the search. A major performance enhancement is *dynamic programming*. Since the same states in the MDP can often be reached via more than one sequence of actions, we cache the expected rewards and corresponding policies of the visited states and reuse it when possible. In addition we use the following pruning techniques.

4.0.1 Sibling-Class Pruning

In this section we introduce the Sibling-Class Theorem (SCT) and its use to prune the search tree. This theorem states that in some cases, the optimal order for executing actions can be determined directly without search. It was proved in [9] in the context of "probabilistic AND-OR tree resolution" (PAOTR). In [5], the AND part of the theorem is proven in the context of simplified attack graphs. Both works assume that actions have no preconditions, i.e., the inner nodes of the AND/OR tree represent only the conjunctions and disjunctions and do not have any costs or probabilities of success. Moreover, the theorem was proven only for the special case of trees not for directed acyclic graphs. We generalize the SCT to handle the more general case that we need for our attack graphs.

Actions $a,b \in A$ belong to the same *OR-sibling class* iff they have the exact same effects. Actions $\alpha \subseteq A$ belong to the same *AND-sibling class* iff there is a "grandchild" action $g \in A$ that can be executed iff all of the actions in α are successful and none of the actions has an effect that would help enable other actions, besides g. We define the *R-ratio* of actions in sibling classes as R(a) = $\frac{p_a}{c_a}$ if a belongs to an OR-sibling class; and R(a) = $\frac{1-p_a}{c_a}$ if a belongs to an AND-sibling class.



(a) Local+*i* network (b) Chain-*i* network (c) Main-*i* network

Figure 2: Network topologies used in the experiment section.

THEOREM 1 (SIBLING CLASS THEOREM). *Let ξ be the optimal attack policy for the attack graph AG. Then for any actions x, y from the same sibling class, such that $R(y) > R(x)$, x is never performed before y in ξ.*

Intuitively, in OR-sibling class, it is reasonable to start with the high-probability low-cost actions. In AND-sibling class, it is reasonable to start with cheap actions that have low success probability to fail fast and avoid paying unnecessary costs. The theorem is formally proven in the extended paper. [5]

Unfortunately, actions that can interact with honeypots violate the monotonicity property assumed in the proof of SCT. These actions cannot be pruned neither preferred to other actions. Thus, we prune only actions that do not interact with honeypots, belong to exactly one sibling class and there is another action in the same sibling class with higher R-ratio.

4.0.2 Branch and Bound

We compute lower and upper bounds of the expected reward in each MDP state. Consequently, we use them to prune the MDP subtrees if they are provably suboptimal.

Lower bounds (LB) are computed from the values of the previously computed MDP subtrees and bound the future MDP subtrees. First, we present necessary properties of the optimal attack policy.

PROPOSITION 1. *Let ξ be the optimal attack policy in state s starting with action a, ξ_+ and ξ_- be its (sub)policies if action a succeeds or fails, respectively, and $\mathbb{E}[\xi]$ be the value of the policy. Then, (i) $\mathbb{E}[\xi_+] + \rho - c_a/p_a \geq \mathbb{E}[\xi_-]$, where $\rho = \sum_{f:\mathrm{eff}(a)\setminus\phi} r_f$ is an immediate reward and (ii) $\mathbb{E}[\xi] \geq \mathbb{E}[\xi_-]$.*

In Fig. 1d is the first decision point for the attack graph in Fig. 1a, where the attacker decides among three available actions: (*Exploit-Firewall*, *Send-MW-Email* or *Create-Dictionary*). In every decision point of the MDP's depth-first search tree we explore each of its subtrees one by one and use maximal value M of the previous actions as a lower bound value for next action. Since every action's outcome (success or fails) is explored separately, we bound them individually. Action a's successful branch is bounded by $M - \rho + c_a/p_a$, which results from Prop. 1(i). Its false branch is bounded by $\frac{M+c_a-p_a(\mathbb{E}[a^+]+\rho)}{1-p_a}$, where $\mathbb{E}[a^+]$ is the action a's successful branch's expected reward, which results from Prop. 1(ii).

We compute the **upper bound** (UB) as the expected reward of the optimal policy in an attack graph relaxed in two ways. First, we ignore the action costs ($c_a = 0$) and the probabilities of touching HPs ($h = 0$). This only improves the expected utility of the attack and causes the optimal strategy to always use all available actions. Therefore, we can compute the expected reward of the policy by computing the probability that each fact is achieved if the attacker attempts all actions. In order to compute this probability efficiently, we run a depth first search in the attack graph from each fact which provides a reward, traversing edges only in the opposite direction. Moreover, to avoid complications caused by cycles, we disregard edges leading to AND nodes from already traversed parts of the attack graph. Removing edges to AND nodes can always only increase the probability of reaching the reward fact.

[5] available at `http://goo.gl/j1y9Lv`.

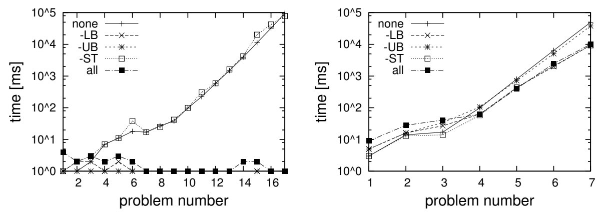
5. EXPERIMENTS

The experiments analyze the algorithm for optimal attack policy computation and evaluate game models for network hardening problems. All experiments were run in single thread on 8-core Intel i7 3.5GHz processor with 10GB memory limit. Attack graphs for the experiments were generated with MulVAL [25] tool, augmented with additional cost and probability information.

5.1 Optimal Attack Planning

In the following experiments we examine the optimal attack policy computation algorithm on following three network topologies: (i) Local-*i* network depicted in Fig. 1c, (ii) Local+*i* network depicted in Fig. 2a and (iii) Chain-*i* network depicted in Fig. 2b. Parameter *i* scales each networks topology in a different dimension: (i) in Local-*i* network, parameter *i* denotes the number of client vulnerabilities in a workstation, (ii) in Local+*i* network, parameter *i* denotes the number of workstations with same connectivity but different vulnerability and (iii) in the Chain-*i* network, parameter *i* denotes the number of pair workstations, where each pair has different both, the connection in the network and a vulnerability. Host types are labeled with (r,l,c), where r,l,c denotes the numbers of *remote*, *local* and *client* vulnerabilities of that type. The topology of the network and the number and type of vulnerabilities determines the number of actions in the attack graph.

5.1.1 Pruning Techniques



(a) Local-*i* (b) Local+*i*

Figure 3: Time comparison of the pruning techniques. Legend: none - no pruning technique, (-LB) - all but lower bound, (-UB) - all but upper bound, (-ST) - all but Sibling Theorem, (all) - all pruning techniques.

In this section we evaluate contribution of the pruning techniques: Sibling Theorem (ST), lower bound (LB), and upper bound (UB) for the branch and bound. Since techniques may prune the same branches, we measure each technique's contribution by measuring the algorithms slowdown after disabling it. Thus, we compare 5 settings: *none* (no techniques is used), *all* (all techniques are used), *-ST* (all but ST), *-LB* (all but LB) and *-UB* (all but UB). Fig. 3 shows optimal attack policy computation times of networks Local-*i* and Loca+*i* using each technique setting. In Local-*i* network we add vulnerabilities to a workstation which creates large decision points. However, since all vulnerabilities have the same effect, to compromise the workstation, they belong to the same OR-sibling class which is effectively pruned by the sibling-theorem and the optimal action is selected without computation (compare *all* to *-ST*). In Local+*i* network the branch and bound helps the most (*-LB* and *-UB*). The results for the Chain-*i* network were very similar to the results for the Local+*i* network, thus, we did not include them here. In the remaining of experiments, we use all techniques since they do not have negative impact and can dramatically speed up the computation. We refer to the proposed attack planning algorithm as DP (dynamic-programing) algorithm.

5.1.2 Comparison with Other Algorithms

We compare DP to other two approaches that can compute the optimal policy for an attack graph. The first method converts the AG it to an *Unconstrained Influence Diagram* (UID) as described in [16] and uses GUIDO [15] to solve it. The second approach translates AG to a probabilistic planning domain problem, which is then solved by SPUDD [11]. It uses iterative value computation and was 3rd best planner in 2011 at the International Planning Competition (IPC) in the MDP track. We chose SPUDD because it guarantees to return an optimal contingency policy, while the first two planners from IPC do not. In Tab. 1(a) we present computation times of the algorithms. Our algorithm dramatically outperforms the other two approaches, where the hardest network Local+8 was solved 5x faster than using SPUDD planner.

Problem	# A	DP	UID	SPUDD
Local-3	9	**<1**	**<1**	**<1**
Local-11	25	**<1**	(OoM)	3
Local-19	41	**<1**	(OoM)	3348
Local+3	16	**<1**	71	1
Local+6	28	**3**	(OoM)	125
Local+8	36	**406**	(OoM)	1951
Chain-3	20	**<1**	21,68	7
Chain-5	32	**<1**	(OoM)	2646
Chain-7	44	**<1**	(OoM)	Err

(a)

host	1	2	3	5
db	0	0	1	2
srv	1	1	1	2
vpn	0	1	1	1
1grp	0	0	0	0
2grp	0	0	0	1

(b)

Table 1: (a) Computation times (in seconds) of computing optimal attack policies by DP, GUIDO and SPUDD. (OoM) - Out of Memory, (Err) - planner exited with segmentation error. (b) Optimal allocation of different number of HPs (columns) to the host types (rows) in the Main-7 network.

5.2 Network Hardening Game Analysis

In this section we evaluate the network hardening game. We focus on a typical small business network topology Main-*i* depicted in Fig. 2c, where parameter *i* denotes the total number of host types in the network. Main-*i* network consists of a Server (srv), a Firewall (fw), a Database (db), a vpn, a group of 20 workstations (1grp) and it is extended by an additional 4 workstation host types (*n*grp) restricted by parameter *i*. On this network we find the honeypot deployment that minimizes the defender's loss and analyse the results. Attacker's actions costs in AG are set randomly between 1 and 100. The rewards for gaining the root privileges (r_a) to the host types are 1000 for workstations (ws), 2000 for vpn, 2500 for server (srv) and 5000 for database (db). For simplicity, we assume that the defender's loss (l_t) is equals to the attacker reward. However, this payoff restriction is not required in our model. The defender pays $c(t) = \gamma l_t$ for adding honeypot of type t, where $\gamma \in \mathbb{R}^+$ is parameter we alter. It corresponds to the fact that more valuable hosts are generally more costly to imitate by honeypots.

Scalability Experiment.

We analyze the scalability of our game model to determine the size of the networks that can reasonably be solved using our algorithms. For each network Main-*i*, we create a game model and find Stackelberg equilibrium. First, we present computation time comparison for solving a game with and without use of cached within a game in Fig. 4a. When MDP trees are computed with various HP allocation settings, often the same MDP subtrees are reached; caching and reusing their values avoids the unnecessary repeated computation. Next, in Fig. 4b we present computation times (note log-scale y-axis) to solve networks Main-6 through Main-8 (individual plots), given the number of HPs that the defender can deploy. The algorithm scales exponentially with both parameters. However, we note that typical real-world cases will have few honeypots.

5.2.1 Case-Study

In this section we examine in detail the network Main-7 with $\gamma = 0.02$. We analyze the defender's loss dependence on the number of honeypots and his relative regret for modeling the attack graph inaccurately (e.g., if he estimates the action costs, action probabilities or rewards incorrectly).

Defender's loss.

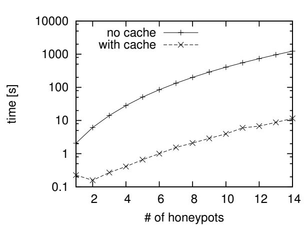
Using a large number of HPs is not necessarily beneficial for the defender as they may cost more than they contribute to protecting the network. The defender's expected loss using different number of the HPs is shown in Fig. 4c (see "optimal, $\gamma = 0.02$"), where it is optimal to deploy 6 HPs for $\gamma = 0.02$ and for 9 HPs for the cheaper $\gamma = 0.01$ setting. In the same figure, we also present the defender's loss for deploying random honeypots instead the optimal suggested by our model, which is roughly twice as bad. We also note how dramatically merely 3 HPs can decrease the expected loss using the game-theoretic approach.

In Fig. 4d are separate parts of the defender's loss: the expected loss for having the hosts attacked and the costs for deploying the HPs. This kind of analysis is useful if other restrictions are posed on the defender, e.g., limited budget expenses on hardening the network. For instance, in our case it shows that purchasing the deploying fourth honeypot will not increase the network security compared to deployment of three honeypots.

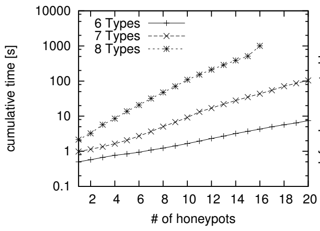
In Tab. 1(b) we present the defender's optimal HP types allocations (rows) for a given total number of deployable HPs (columns). I.e., with one HP, it is best to duplicate the server (srv). The server is not the most valuable type (2500 while the database is worth 5000). However, it is a bottleneck of the network; protecting the server partially defends both the server and the database, rather than only database. With two HPs it is best to duplicate the server and VPN (again, the bottlenecks). Only with the 3rd HP does the defender duplicate the database.

Sensitivity Analysis.

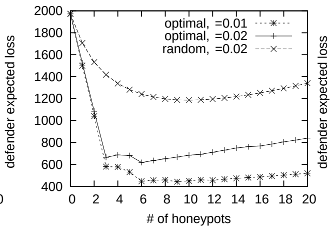
Computing the defender's optimal strategy heavily relies on the Attack Graph structure and knowledge of the action costs, probabilities and rewards, which the defender can only approximate in



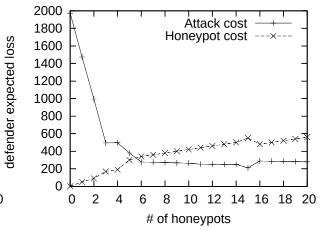
(a) Cache vs no-cache



(b) Scalability



(c) Defender's loss



(d) Defender's costs

Figure 4: (a) With and without cache computation times comparison. (b) Time comparison of solving the game for various networks and the number of HPs; (c) Defender's expected loss given the γ and allocating HPs optimally and randomly. (d) Individual parts of the defender's loss: the expected cost of having the network attacked (Attack cost) and cost of deploying the HPs (Honeypot cost).

a real-world scenarios. In the following experiments we analyze the sensitivity of the defender's strategy and utility to inaccurate approximation of the attack graphs.

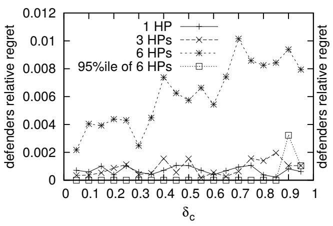
We use the following notation: defender's strategy $\bar{d}$ (resp. loss $\bar{l}_{\bar{d}}$) is computed from the attack graph known to the defender, which imprecisely approximates the attack graph truly describing attacker's behavior; d (resp. loss l_d) is the optimal strategy based on the true attack graph according which the attacker really behaves; and $l_{\bar{d}}$ is the defender's loss when strategy $\bar{d}$ is used against the real attack graph.

In the sensitivity analysis we compute the *relative regret* as $|l_{\bar{d}} - l_d|/\bar{l}_{\bar{d}}$, which is the defender's relative loss ratio for not knowing the attacker's real attack graph. In other words, if the defender knew the true attack graph model and acted optimally, this is how much he would gain. Thus, $regret = 0$ means he has no regret and $regret = 1$ means that his loss is by 100% greater than it could have been.

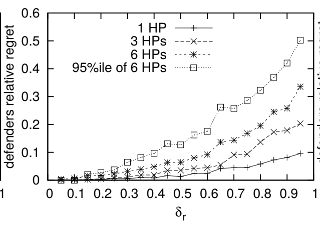
In this experiments the attack graph action probabilities and costs were chosen randomly from the intervals $p_a \in [0,1]$, $c_a \in [0,200]$ in order not to depend on the specific initial attack graph values provided by MulVAL. To generate the imprecise attack graph available to the defender, we perturbed the generated attack graph based on values $(\delta_p, \delta_c, \delta_r) \in [0,1]^3$, where each value represents the limit on the size of the perturbation on action probabilities (δ_p), costs (δ_c) and fact rewards (δ_r). The perturbed attack graph is obtained as follows: for each action $a \in A$ the perturbed probability $\bar{p}_a$ is chosen from an interval $[p_a - \delta_p, p_a + \delta_p]$ restricted to $[0.05, 0.95]$ to prevent them becoming impossible ($\bar{p}_a = 0$) or infallible ($\bar{p}_a = 1$); perturbed cost $\bar{c}_a$ is chosen from $[c_a(1-\delta_c), c_a(1+\delta_c)]$; and for each fact $f \in F$, the perturbed fact reward is $\bar{r}_f \in [r_f(1-\delta_r), r_f(1+\delta_r)]$, where the values are selected uniformly within the given intervals. Action probabilities are perturbed absolutely (by $\pm\delta_p$), but the costs and rewards are perturbed relative to their original value (by $\pm\delta_c c_a$ and $\pm\delta_r r_f$). The intuition behind this is that the higher the cost or reward values the larger the errors the defender could have made while modeling them, which cannot be assumed for probabilities.

In our experiments we perturbed the each component from 0.05 to 0.95 by 0.05 steps and measured the defender's loss. The results presented are mean values computed from 100 runs. In Fig. 5 we present the defender's relative regret for increasing error perturbations of each component individually in Fig. 5a-c and altogether in Fig. 5d for deploying 1, 3 and 6 HPs. In our experimental setting, the defender's strategy was the least sensitive to the action cost perturbations (Fig. 5c), where the relative regret reaches only 0.002 for $\delta_c = 0.95$ for 6 HPs. Sensitivity to reward perturbations (Fig. 5b) reached relative regret about 0.3 for the worst case $\delta_r = 0.95$. Finally, the most sensitive it is to the probability perturbation (Fig. 5c), where relative regret reached almost value 0.8 for $\delta_p = 0.95$. Fig. 5d depicts relative regret for perturbing all three components, reaching about 1.2 for the worst case. Overall, results show that within a reasonable perturbation boundaries, the algorithm is robust. I.e., assuming that the defender models all components imprecisely by at most 30% ($\delta_c = \delta_p = \delta_r = 0.3$ in Fig. 5d) in scenario with 6 HPs, his HP deployment choice could have been bettered off by only 17%. While imprecision of 30% is quite large in our perturbation definition, the results in a quite little regret.

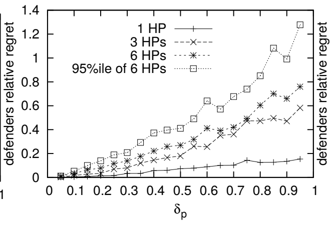
Finally, note that with an increasing number of HPs, the defender's relative regret grows (e.g., Fig. 5), however, this can be misleading. Further data analysis revealed that the actual *absolute regrets* $|l_{\bar{d}} - l_d|$ for 3 HPs (318) and 6 HPs (340) are very similar. The reason why relative regret seem to grow with the increasing number of HPs is due to the decrease of the observed loss $\bar{l}_{\bar{d}}$ (denominator it relative regret formula). The observed loss $\bar{l}_{\bar{d}}$ for 3 HPs is 1994, while for 6 HPs 1049, which shows that the relative regret does not seem to depend on the number of HPs.



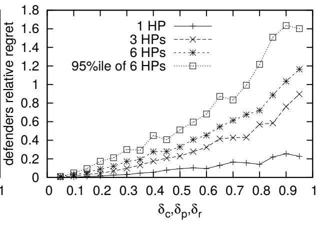
(a) Defender's loss



(b) Defender's costs



(c) Scalability



(d) Sensitivity

Figure 5: Defender relative regret from perturbation only action costs (a), only action probabilities (b), only action rewards (c) or all three components (d) in networks with 1, 3 and 6 HP, and 95th percentile for 6 HPs.

6. DISCUSSION

Lately, the game theoretic approaches are becoming more and more used in the security domains as they presents a powerful tool to the model adversaries counter-strategies to the defender's actions. Although, the assumption that the adversary acts fully rationally—as assumed in this paper—might be too strong, the proposed framework can be used to model a whole variety of the more realistic adversaries.

Bounded rationality is often used to model a more human-like deciding agents that do not act fully rationally. One of the bounded rationality concepts is used in a *quantal response* [17] equilibrium, where players are assumed to make errors in choosing the strategy. However, they are likely to make error that cost them little, while very costly errors are unlikely. *Cognitive Hierarchy Theory* [6] grasps the human thinking approach differently. It has inductively defined strategic categories as follows: 0-level players randomize uniformly, 1-level players play best response assuming that other players are of 0-level, etc., n-level players play best response assuming the other players are of $(n-1)$-level. It imitates the player recursive reasoning about the other players reasonings and has been shown to be consistent with observations of human behavior.

Natural extension of this work is to relax the assumption that the attacker learns the honeypot types before he attacks. It leads to *imperfect information* games where the attacker observes a network, however, he does not know the original network topology before introducing honeypots neither the defender's honeypot allocation action. In this game both player's might have a prior knowledge about the typicality network topologies that defender hardens and attacker attacks (typical networks in an organization, companies, etc.). It is a type of *deception game*, where defender acts to conceal the information of the honeypot location for the adversary as much as possible.

Finally, *Bayesian games* are often used in domains, where player's do not know exactly the other player's payoffs (their *types*). It can be used to model adversaries with different target hosts they want to compromise. The defender knows only the probability distribution over the attacker types and he acts to maximize his expected utility.

7. CONCLUSION

We introduce a game-theoretic model for the network hardening problem. The defender seeks an optimal deployment of honeypots into the network, while the attacker tries to attack the network and avoid the interaction with the honeypots. Our model provides a novel combination of using compact representation of the strategies of the attacker in the form of attack graphs, and using deception by the defender. By translating the attack graphs into MDPs and employing a number of pruning techniques, we are able to solve problems of realistic size and analyze the results for realistic case studies. Moreover, we showed that our model produces robust solutions even if the input data are imprecise.

Our work has significant potential for further research. Since the majority of the required input data can be automatically acquired by standard network scanning tools, or extracted from existing vulnerability databases, the proposed model can be deployed in real-world networks and evaluated in practice. Secondly, our model can be further extended from the game-theoretical perspective and use additional uncertainty about the knowledge of the attacker, or model multiple types of the attacker using Bayesian variants of Stackelberg games.

Acknowledgments

This research was supported by the Office of Naval Research Global (grant no. N62909-13-1-N256) and Danish National Research Foundation and The National Science Foundation of China (under the grant 61361136003) for the Sino-Danish Center for the Theory of Interactive Computation. Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the programme "Projects of Large Infrastructure for Research, Development, and Innovations" (LM2010005), is greatly appreciated. Viliam Lisý is a member of the Czech Chapter of The Honeynet Project.

8. REFERENCES

[1] P. Ammann, D. Wijesekera, and S. Kaushik. Scalable, graph-based network vulnerability analysis. In *CCS*, pages 217–224, 2002.

[2] E. Bacic, M. Froh, and G. Henderson. Mulval extensions for dynamic asset protection. Technical report, DTIC Document, 2006.

[3] R. Bellman. Dynamic programming and lagrange multipliers. *PNAS*, 42(10):767, 1956.

[4] M. S. Boddy, J. Gohde, T. Haigh, and S. A. Harp. Course of action generation for cyber security using classical planning. In *ICAPS*, pages 12–21, 2005.

[5] A. Buldas and R. Stepanenko. Upper bounds for adversaries' utility in attack trees. In *GameSec*, pages 98–117. 2012.

[6] C. F. Camerer, T.-H. Ho, and J.-K. Chong. A cognitive hierarchy model of games. *The Quarterly Journal of Economics*, pages 861–898, 2004.

[7] G. Cohen, M. Meiseles, and E. Reshef. System and method for risk detection and analysis in a computer network, Jan. 17 2012. US Patent 8,099,760.

[8] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *EC*, pages 82–90, 2006.

[9] R. Greiner, R. Hayward, M. Jankowska, and M. Molloy. Finding optimal satisficing strategies for and-or trees. *Artificial Intelligence*, pages 19–58, 2006.

[10] R. A. Grimes, A. Nepomnjashiy, and J. Tunnissen. Honeypots for windows. 2005.

[11] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. Spudd: Stochastic planning using decision diagrams. In *UAI*, pages 279–288, 1999.

[12] J. Homer, X. Ou, and D. Schmidt. A sound and practical approach to quantifying security risk in enterprise networks. *Kansas State University Technical Report*, pages 1–15, 2009.

[13] J. Homer, S. Zhang, X. Ou, D. Schmidt, Y. Du, S. R. Rajagopalan, and A. Singhal. Aggregating vulnerability metrics in enterprise networks using attack graphs. *Journal of Computer Security*, pages 561–597, 2013.

[14] K. Ingols, R. Lippmann, and K. Piwowarski. Practical attack graph generation for network defense. In *ACSAC*, pages 121–130, 2006.

[15] J. Isa, V. Lisy, Z. Reitermanova, and O. Sykora. Unconstrained influence diagram solver: Guido. In *IEEE ICTAI*, pages 24–27, 2007.

[16] V. Lisý and R. Píbil. Computing optimal attack strategies using unconstrained influence diagrams. In *PAISI*, pages 38–46. 2013.

[17] R. D. McKelvey and T. R. Palfrey. Quantal response equilibria for extensive form games. *Experimental economics*, 1(1):9–41, 1998.

[18] P. Mell, K. Scarfone, and S. Romanosky. Common vulnerability scoring system. *Security & Privacy*, pages 85–89, 2006.

[19] S. Noel and S. Jajodia. Managing attack graph complexity through visual hierarchical aggregation. In *Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security*, pages 109–118. ACM, 2004.

[20] S. Noel and S. Jajodia. Optimal ids sensor placement and alert prioritization using attack graphs. *Journal of Network and Systems Management*, pages 259–275, 2008.

[21] S. Noel, S. Jajodia, L. Wang, and A. Singhal. Measuring security risk of networks using attack graphs. *International Journal of Next-Generation Computing*, 1(1):135–147, 2010.

[22] J. Obes, C. Yamada, and G. Richarte. System and method for extending automated penetration testing to develop an intelligent and cost efficient security strategy, July 16 2013. US Patent 8,490,196.

[23] J. L. Obes, C. Sarraute, and G. Richarte. Attack planning in the real world. *arXiv preprint arXiv:1306.4044*, 2013.

[24] X. Ou, W. F. Boyer, and M. A. McQueen. A scalable approach to attack graph generation. In *CCS*, pages 336–345, 2006.

[25] X. Ou, S. Govindavajhala, and A. W. Appel. Mulval: A logic-based network security analyzer. In *USENIX Security*, 2005.

[26] N. Provos. A virtual honeypot framework. In *USENIX Security Symposium*, volume 173, 2004.

[27] M. T. Qassrawi and Z. Hongli. Deception methodology in virtual honeypots. In *Networks Security Wireless Communications and Trusted Computing (NSWCTC), 2010 Second International Conference on*, volume 2, pages 462–467. IEEE, 2010.

[28] E. Rasmusen and B. Blackwell. Games and information. *Cambridge, MA*, 15, 1994.

[29] A. Rubinstein. *Modeling bounded rationality*, volume 1. MIT press, 1998.

[30] R. E. Sawilla and X. Ou. Identifying critical attack assets in dependency attack graphs. In S. Jajodia and J. Lopez, editors, *Computer Security - ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 18–34. Springer Berlin Heidelberg, 2008.

[31] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. Wing. Automated generation and analysis of attack graphs. In *IEEE S&P*, pages 273–284, 2002.

[32] M. Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[33] M. Tambe. *Security and game theory: Algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.

APPENDIX

A. PROOFS

A.1 Propositions and Lower Bounds

In this section we proof used propositions and lower bounds. We begin with the necessary propositions.

PROPOSITION 2. *Let ξ be an optimal attack policy starting with action a, $\mathbb{E}[\xi]$ be the expected cost of the policy ξ and $\mathbb{E}[\xi_+]$ (resp. $\mathbb{E}[\xi_-]$) be the expected cost of subpolicy ξ after action a succeeds (resp. fails). Then $\mathbb{E}[\xi] \geq \mathbb{E}[\xi_-]$.*

PROOF. The first part we prove by contradiction. Assume that $\mathbb{E}[\xi] < \mathbb{E}[\xi_-]$. Then, due to the monotonicity property the attacker could have followed the (sub)policy ξ_- before performing action a, which would have saved him the cost of the action c_a. This new policy would have had higher expected value than the policy ξ, which contradict the optimality of ξ. □

PROPOSITION 3. *Let ξ be the optimal attack policy starting with action a, $\mathbb{E}[\xi]$ be the expected cost of the policy ξ and $\mathbb{E}[\xi_+]$ (resp. $\mathbb{E}[\xi_-]$) be the expected cost of subpolicy ξ after action a succeeds (resp. fails). Then $\mathbb{E}[\xi_+] + \rho - c_a/p_a \geq \mathbb{E}[\xi^-]$, where $\rho = \sum_{f:f_a\setminus\phi} r_f$ is an pure immediate reward received if action a succeeds.*

PROOF. For convenience, we denote $\bar{p_a} = 1 - p_a$.

$$\begin{aligned}
\mathbb{E}[\xi] &= -c_a + p_a(\mathbb{E}[\xi_+] + \rho) + \bar{p_a}\mathbb{E}[\xi_-] \\
\text{Prop. 2} \Rightarrow \mathbb{E}[\xi_-] &\leq -c_a + p_a(\mathbb{E}[\xi_+] + \rho) + \bar{p_a}\mathbb{E}[\xi_-] \\
p_a\mathbb{E}[\xi_-] &\leq -c_a + p_a(\mathbb{E}[\xi_+] + \rho) \\
\mathbb{E}[\xi_-] &\leq -c_a/p_a + \mathbb{E}[\xi_+] + \rho.
\end{aligned}$$

□

A.1.1 Lower Bounds

In a decision point, the MDP search explores subtrees of all applicable actions one by one. As each action subtree's expected reward is computed, we can their results to bound the future actions' subtrees.

PROPOSITION 4. *Assume the actions $a_1, \ldots, a_{k+1} \in A$ to be in the same decision point in state $s = (\alpha, \phi, \tau)$ of the depth-first search tree for MDP from which the subtrees of actions $a_1, \ldots, a_k$ have been explored with their maximal expected reward M. Let ξ be the optimal attack policy starting with action a_{k+1}, $\mathbb{E}[\xi]$ be the expected cost of the policy ξ and $\mathbb{E}[\xi_+]$ (resp. $\mathbb{E}[\xi_-]$) be the expected cost of subpolicy ξ after action a succeeds (resp. fails).*

Strategy ξ starting with action a_{k+1} and followed optimally has higher expected reward then strategies starting with actions $a_1, \ldots,$ a_{k+1} (and followed optimally) iff

$$\mathbb{E}[\xi_+] > M - \rho + \frac{c_{a_{k+1}}}{p_{a_{k+1}}}$$

and

$$\mathbb{E}[\xi_-] > \frac{M + c_{a_{k+1}} - p_{a_{k+1}}(\mathbb{E}[\xi_+] + \rho)}{1 - p_{a_{k+1}}}.$$

The proof of the first inequality:

PROOF.

$$\begin{aligned}
\mathbb{E}[\xi] = -c_{a_{k+1}} + p_{a_{k+1}}(\mathbb{E}[\xi_+] + \rho) + \bar{p_a}\mathbb{E}[\xi_-] &> M \\
-c_{a_{k+1}} + p_{a_{k+1}}(\mathbb{E}[\xi_+] + \rho) + p_{\bar{a}_{k+1}}(\mathbb{E}[\xi_+] + \rho - \frac{c_{a_{k+1}}}{p_{a_{k+1}}}) &> M \\
\mathbb{E}[\xi_+] &> M - \rho + \frac{c_{a_{k+1}}}{p_{a_{k+1}}}.
\end{aligned}$$

□

The proof of the second inequality:

PROOF.

$$\begin{aligned}
\mathbb{E}[\xi] &> M \\
-c_a + p_a(\mathbb{E}[\xi_+] + \rho) + p_{\bar{a}_{k+1}}\mathbb{E}[\xi_-] &> M \\
\mathbb{E}[\xi_-] &> \frac{M + c_{a_{k+1}} - p_{a_{k+1}}(\mathbb{E}[\xi^+] + \rho)}{p_{\bar{a}_{k+1}}}.
\end{aligned}$$

□

ENGMAS – Understanding Sanction under Variable Observability in a Secure Environment

Hongying Du
NC State University
Raleigh, NC, United States
hdu2@ncsu.edu

Bennett Narron
NC State University
Raleigh, NC, United States
bynarron@ncsu.edu

Nirav Ajmeri
NC State University
Raleigh, NC, United States
najmeri@ncsu.edu

Emily Berglund
NC State University
Raleigh, NC, United States
emily_berglund@ncsu.edu

Jon Doyle
NC State University
Raleigh, NC, United States
jon_doyle@ncsu.edu

Munindar P. Singh
NC State University
Raleigh, NC, United States
mpsingh@ncsu.edu

ABSTRACT

Norms are a promising basis for governance in secure, collaborative environments—systems in which multiple principals interact. Yet, many aspects of norm-governance remain poorly understood, inhibiting adoption in real-life collaborative systems. This work focuses on the combined effects of sanction and observability of the sanctioner in a secure, collaborative environment. We introduce ENGMAS (Exploratory Norm-Governed MultiAgent Simulation), a multiagent simulation of students performing research within a university lab setting. ENGMAS enables us to explore the combined effects of sanction (group or individual) with the sanctioner's variable observability on system resilience and liveness. The simulation consists of agents maintaining "compliance" to enforce security norms while also remaining "motivated" as researchers. The results show with lower observability, agents tend not to comply with security policies and have to leave the organization eventually. Group sanction gives the agents more motive to comply with security policies and is a cost-effective approach comparing to individual sanction in terms of sanction costs.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*multiagent systems*

General Terms

Security, Human Factors

Keywords

Norms, Resilience, Liveness, Safety, Observability, Multiagent System

1. INTRODUCTION

Secure, collaborative environments often suffer from *ad hoc* implementations of policy developed by its governing principal through careful scrutiny of secure practices and reactionary measures to threat [5]. However, assessing the security of an environment is costly, and while having the clarity of hindsight, patching existing vulnerabilities does not ameliorate the damages caused by an attack. To address this issue, norm-based solutions for governance have emerged to introduce a more pliable means of developing policy [7, 14, 15]. While promising, attempts-to-date have failed to fully capture a low-level interpretation of a norm-governed system. Still, the pursuit of understanding the application of norms to system security has revealed a promising frontier for research.

The ultimate goal of our research is to draw nearer to a complete mathematical model of norms, so that we may understand their behavior as a form of governance in a system of heterogeneous, autonomous principals. More specifically, we are concerned with how these norms and policies may influence a system's liveness and resilience; however, we are still posed with the question: Can one predict the liveness and resilience of a system as a function of its norms and policies and the social environment? Further, do any trade-offs exist between liveness and resilience that may be influenced by norms and policies?

The answers to these questions are computationally hard and are, therefore, outside of our ability to address using a game-theoretic approach. As a result, we have designed and implemented an exploratory multiagent simulation, called ENGMAS, to address our questions. The simulation is a centralized system wherein autonomous agents perform tasks under the scrutiny of a governing principal. ENGMAS is regulated by multiple, adjustable settings, many of which may have profound effects on the outcome of the simulation; however, the primary focus of this paper is to explore the variable observability of the governing principal and the type of sanction type (individual or group) applied to norm violation.

We acknowledge two factors as possible influence on the resilience and liveness of a system: (1) a sanctioning agent's observability of its environment and (2) the means of sanction applied (group or individual).

2. TERMINOLOGY

The variability in the literature regarding the terms we use to describe our research warrants careful attention. Thus, we devote this section to defining our interpretations of each concept we intend to present.

2.1 Norms and Sanction

We recognize the term, *norms*, to describe "directed normative relationships between participants in the context of an organization" [14]. For example, consider a university setting in which there exist multiple research labs nested within its various departments. Within each lab, there is a finite set of graduate students assisting with departmental research, utilizing security-critical artifacts (namely, PCs or machines) connected to a shared network. The network may be monitored by IT staff, whose role it is to ensure the safety of the system. In doing so, the network administrator may implement a directed set of agreements (including, but not limited to, patching the operating system, updating passwords, and maintaining a firewall) which must be met by each of the graduate student researchers in order to avoid some consequence. Alternatively, graduate students are also tasked with completing research in order to maintain grants, earn credit hours, or other obligations. Each student's advisor has an expectation of motivation directed towards the student.

Failure to comply with normative expectations is met with *sanction* (consequence for norm violation applied to a principal or group of principals) by a sanctioning agent (e.g., the IT staff in the previous example). A sanction may be positive or negative and manifested in reprimand or reward, respectively [11]; though, in the context of the university research lab environment, norm violations would result in negative sanction. When an individual principal is singled-out and censured for defecting against a norm, we recognize this as "individual" sanction. Alternatively, when sanction is applied to a group of individuals for the actions of some subset of that group, we recognize this as "group" (also known as "collective") sanction [8].

2.2 Safety and Liveness

Lamport is credited with the earliest acknowledgment of the safety and liveness compromise [9]. Following from his seminal work, we define *safety* as the stipulation that some action (or inaction) does not cause the system to enter a bad state. In our university lab example, we describe the safety concerns of the system administrator as his or her desire to maintain a secure system. The safety of the system is in direct correlation with the magnitude of security compliance by the graduate students in the lab. That is, if the students forego the norms that describe the security measures they are expected to perform, the system is likely to enter a compromised state. In response, the administrator may shut down some subset of network connections that he or she deems responsible for the threat, an action that would directly benefit the desires of the administrator but would be devastating to the system's liveness.

Liveness is characterized by maintaining a good state, or more specifically, a state of production [2]. Counter to the safety example, liveness can be inhibited by safety precautions and serves the interest of the research advisers. Both enacting safety measures and conducting research are costly endeavors, and limitation of resources (i.e., time) will almost certainly result in the investment of one obstructing the ability to perform the other. In the lab, we would witness students attempting to adhere to security policy at the expense of research load, or vice versa.

2.3 System Resilience

Multidisciplinary interpretations of *system resilience* (in Economics, Anthropology, Social-Ecology, and Resilience Engineering, among others) abstract to several meaningful definitions, each with respect to its own application (e.g., [1, 4, 6, 12, 13]). For the purpose of our research, we align our understanding of resilience with Sheridan, who describes it as the ability to "recover and restore the system to the original state or, if need be, some acceptable state that is different but still safe" [13]. We do not attempt to define the notion of an "acceptable" state and further recognize it to be an arbitrary factor. The acceptability of a state is subject to the opinion of any active principal in the system, and more often than not, these views are conflicting. If we consider the dichotomy presented earlier between the expectations of the system administrator and those of a graduate student adviser, there is no guarantee that there is any satisfiable union between any set of states that both parties would deem "acceptable". For this reason, our metric for resilience will be determined by a variable dependent on both safety and liveness, named *load*.

Load is defined by the ratio of the actual amount of work completed against the total potential, aggregated over the sum of the number of active, performing agents and the number of the non-active agents in the system. It is representative of the productivity of a system at a single time step. Tasks performed by agents which contribute to the safety of a system are not considered to be "productive", and thus, are not a component in the calculation of load. However, such tasks do effectively assist in maintaining a safe state and allow productive actions to occur. In our lab analogy, performing research would qualify as a productive task, while implementing security features would directly promote the safety of the environment.

Given load, we calculate the resilience of a system by how quickly it recovers from successive disturbances (more specifically, norm violations). This phenomenon is best illustrated by the waxing and waning of load over time. As active principals neglect to fulfill expectations regarding the safety of the system, the resulting downtime will likely quell the productivity of the system. After agent behavior is modified via sanction, a resilient system is likely to efficiently recover to some arbitrarily acceptable state, while a rigid system most likely will not.

2.4 Observability and Efficiency

The primary focus of this paper is to understand system resilience as a result of the combined effects of the sanctioner's observability of his or her environment coupled with the type of sanction applied to principals responsible for norm violation. More explicitly, we are interested in the scenarios of the sanctioner applying individual sanction on identified defectors, or group sanction despite the identities of the defectors under various observabilities.

Each of the aforementioned cases has underlying implications associated with it. For example, the system administrator (i.e, the sanctioner) for our university lab is tasked with sanctioning an agent after he or she realizes that the network has been compromised. If the administrator seeks to identify the specific party that caused the vulnerability, the influence of the individual sanction may not have much affect on the rest of the group. Alternatively, group sanction may lead to vigilance among agents, which could promote whistle-blowing. Further, the more transparent the lab is to surveillance, the more efficiently sanctions may be applied; this may quell the amount of repeated norm violations by a single agent, while partial observability may allow agents to continually ignore normative expectations. We acknowledge these factors as essential to understanding system resilience and have designed a simulation to apply these conditions to a university lab setting.

3. THE SCENARIO

To further investigate our proposed research, we extend our lab scenario to a multiagent simulation, called ENG-MAS. A more formal description of our scenario involves a university graduate research lab (congruent to our running example) and its constituent student researchers, represented by agents. The system contains three types of entities:

(1) A lab, or an organization, wherein student agents perform activities.

(2) A set of Student Agents, each representing a student researcher who controls a PC in the lab. Each agent must respond to the tasks assigned to it using its PC, and will be sanctioned if it violates the norms of the system.

(3) A special, centralized agent named Carlos, who is responsible for applying sanctions to the agents in the lab.

3.1 Carlos

Agents have different duties and, therefore, play different roles within the system. Carlos is in charge of sanctions, thus his responsibilities are to:

- Observe, or monitor the lab's network for any visible norm violations. *Observability (O)* varies on a scale, ranging from 0% (inability to observe norm violations) to 100% (ability to observe all norm violations). For example, if Carlos has observability of 80%, then at each time step there is an 80% chance he will discover a norm violation, independently, given a set of student agents who have failed to comply with security norms.
- Perform sanctions as soon as a norm violation is caught. We represent sanctions as shutting off network access of PCs in the lab for a certain amount of time, which prevents threats to PCs from spreading. After sanction, each student agent whose PC health is below a threshold has to fix its PC. Carlos could perform one of the following two kinds of sanctions:
  - *Individual Sanction*, the PCs controlled by agents who violate norms are disconnected from network for a constant period of time.
  - *Group Sanction*, all PCs in the lab are disconnected from network for a constant period of time, despite the identity of defecting agent.

  There are expenses for Carlos to discover norm violations, which are different for the two kinds of sanctions. For individual sanction, Carlos needs to find out who are the defectors, which costs more than simple observation of norm violations without figuring out the defectors in group sanction.

3.2 Student Agents

Student agents could perform the following actions:

- Domain-related actions, which includes research related actions to fulfill their research responsibility, and security related actions to ensure the security of their PCs and thus to protect the integrity of the lab environment. Security related actions include: patching the operating system, turning the firewall on and off, updating passwords, installing, turning on or off, and updating the anti-virus software.
- Observe, or monitor other student agents' actions. Agents in the lab have the ability to observe the actions of others, though not particularly well. For example, under the constraints of low observability, Carlos may not be capable of observing all activity on the network; thus he may rely on student agents to report norm violations to him. In this case, student agents must be able to observe other agents' or their neighbors' (i.e., agents within close proximity) actions.
- Communication. To report a norm violation to Carlos in the above case, student agents have a channel for communication with Carlos. They may send messages reporting other students for norm violation.

There are two types of tasks (actions that agents are obligated to perform) that may be assigned to student agents:

- A research task represents the research work that a student must finish (for example, writing a paper, reading papers, etc) in order to fulfill the expectations of his or her adviser.
- A security task represents the security precautions a student must take to ensure that his or her PC is safe. It could be one of the security-related actions or any combination of them.

Each task type has two attributes:

- Duration, or the amount of time it takes for a student agent to complete a task. Compared to a research task, a security task is much less time-consuming for an agent, as represented in reality (e.g., doing homework takes much longer than changing a password).
- Deadline, or the amount of time a student agent is allowed to complete a particular task. If it is unable to complete a task by the deadline, the agent's health (for failure of a research task) or PC health (for failure of a security task) will decrease. For each task, deadlines are allotted with ample time for a student to complete it. For research tasks, we determine the deadline by applying a fixed coefficient, c, to the task duration. For example, if a research task consists of five time steps, theoretically an agent is given $5 * c$ time steps to complete it. We consider a time step as the smallest time unit and round decimals to the nearest integer when calculating deadlines.

Each student agent has five attributes:

- *Agent Health* represents a student agent's health and is influenced by the completion status of research tasks assigned to the agent. Agent health is subject to change during the simulation for the following reasons: (1) If an agent finishes a research task before the deadline, its agent health is increased accordingly. (2) If an agent is unable to finish a research task before the deadline, its agent health is reduced accordingly. If an agent's health reaches a fixed low value, it must leave the organization or lab. If an agent leaves, it is no longer able to perform any action and will never return to the simulation and we consider it as non-active.

- *PC Health* represents the PC's health and is influenced by the completion status of security tasks assigned to the agent: (1) If an agent finishes a security task before its deadline, its PC health is increased accordingly. (2) If an agent is unable to finish a security task before its deadline, the agent has committed a norm violation, and its PC health is reduced accordingly. If the PC health drops under a fixed value, the PC has been compromised and is unusable until it is sanctioned by Carlos and starts to fix his PC.

- *Preference* is the probability that an agent will choose to begin working on a research task first, under the condition that it has both a research task and a security task to perform. For example, if an agent's preference is 80% and it has both a research task and a security task on its task list, it will have an 80% chance of considering the research task and 20% chance of considering the security task. Note that by "considering", the agent does not start working on the task. It merely implies that the agent has decided which task to attempt. The probability that an agent begins working on a task is dependent upon one of the following two attributes.

- *Research Motivation* is the probability that an agent will choose to start a research task, given that the agent with both tasks has considered it, or the agent only has a research task. After an agent fails to finish a research task, its research motivation increases, indicating that it is motivated to start a research task early next time.

- *Security Compliance* is the probability that an agent will choose to start a security task, given that the agent with both tasks has considered it, or the agent only has a security task.

Preference, research motivation, and security compliance together dictate which tasks are being started, if any, during a given time step. The initial values of preference, research motivation and security compliance for each student agent are generated via normal distributions.

4. SIMULATION AND EVALUATION

4.1 Assumptions and Settings

Due to the complexity of real-world scenarios, we have made several assumptions in our experiment. We do not trivialize the significance of these variables, nor do we recognize them as arbitrary. In future work, we intend to determine valid replacements through further research, experimentation, and data collection. However, as the nature of our simulation is exploratory, we have intuitively assigned values to these variables, which are held constant throughout all treatments and runs of the simulation. Further, we will use the term, "tick", to denote a single time step or the smallest time unit defined by users. One tick could be of different time length in different applications.

Initial values of agent health and PC health are both 100. Research tasks are assigned every three ticks; security tasks are assigned every seven ticks.

Research tasks are assigned only to active agents (i.e., agents whose agent health are greater than zero), regardless of whether their PCs are down (PC health is zero) or not. For any research task, we generate a number from a normal distribution as its duration. Its deadline is decided by the coefficient and duration as explained before. If an agent's PC is down, it will not be able to perform tasks and its PC health cannot decrease any further. If an agent leaves the lab, its research tasks are cleared, i.e., no more research tasks are on its list.

Security tasks are assigned to active agents with PCs not down. To simplify the simulation, we treat security-related actions all as abstract security tasks and do not distinguish between different security-related actions. For a security task, since it takes relatively less time to finish, we assume each security task actually takes only one tick. We allot the agents a static seven ticks for a security task, since it only takes one tick to complete a security task and assignment is less frequent than research tasks.

For simplicity, we treat all the student agents in the lab as one group, and ignore the implementation of their observation and communication actions. Appendix shows a summary of the parameter values not presented here.

4.2 Runtime Actions

At each tick, the following actions occur:

- New tasks are assigned to the agents in the lab, if possible. We assume a research task may only be assigned to agents who do not already have assigned research tasks with deadlines exceeding the current tick. For example, an agent has a research task with deadline equal to tick 9, then no research task will be assigned to it until tick 10. In this way, we eliminate the possibilities that agents are assigned too many tasks beyond their capabilities. Half of all agents or all available agents, whichever are lower in number, are assigned research tasks. Every agent is eligible to be assigned a security task. An agent may work on at most one task type per tick.

- Student agents attempt to perform tasks. There are two possible status at each tick: the agent is working on an incomplete task or its deadline is not yet expired, so he continues working on the task; or the agent is not currently working on a task, so it must choose a task to perform. In the latter case, there are four extended possibilities:

  (1) If the agent has no task on its list, then it will rest.

  (2) If the agent has only a research task, it chooses whether to do the assigned research task or not, decided by its research motivation.

  (3) If the agent has only a security task, it chooses to do the assigned security task or not, decided by its security compliance.

  (4) If the agent has both the research task and the security task, it makes the decision following this procedure: the agent chooses to consider which task, according to the preference attribute. We call this chosen task the "preferred task". The agent will then choose

whether to start working on the preferred task according to research motivation or security compliance, respectively. If the agent chooses not to start the preferred task, then the agent rests.

If an agent chooses to begin working on a task, we assume the agent will continue to work on it until it is completed, unless the agent leaves the lab, it's being sanctioned, the agent's PC is down, or trying to fix its PC after being sanctioned.

- PC health decays daily proportional to the ratio of the number of agents whose PC health is below 80 over the total number of PCs in the lab. Maximum decrease for a single day is limited to a value of five. This mimics the viral influence of PCs in a bad state on other PCs in the lab, as a network with PCs that have security issues may be more vulnerable and thus influence other PC's health in the network, as well.
- Carlos may or may not observe norm violations, based on observability. As long as he discovers norm violations, he issues a particular sanction depending on his sanction type. We assume a sanction takes one tick. After Carlos issues a sanction, the sanctioned agents' PCs are shut off network access for one tick. After sanction, each agent's security compliance is increased by a fixed percentage (25% in our case), and the agents are forced to begin restoring their PCs to an acceptable state. PC restoration occurs incrementally following sanction, with each agent's PC health increasing by 15% at each time tick until it reaches 80, at which point the agent may resume completing tasks.
- The research motivation and security compliance attributes of each agent decreases under certain conditions in order to mimic the behavior of agent complacency over time. In our simulation, if an agent is not working on a research task and its health is above 60, its research motivation decreases by 0.01 at that tick. Also, if an agent has not been sanctioned for 45 ticks, its security compliance decreases by 0.01. For example, if tick 1 is the last time an agent gets sanctioned, its security compliance starts to decrease at the end of tick 47.

4.3 Metrics

The following metrics are measured over the course of the simulation:

- *Liveness*, or *System Research Motivation (M)*: Calculated as the median value of the averages of all the agents' research motivation values at each tick of the simulation.
- *System Security Compliance (C)*: Similar to system research motivation, calculated as the median value of the averages of all the agents' security compliance values at each tick of the simulation.
- *System Load*: Load is calculated as the ratio of the number of agents who are actively performing a research task at each time tick over the sum of the number of agents who have research tasks on their lists and the number of non-active agents. It measures the percentage of research load by all agents capable of production. We define "System Load" as the median load over the entire simulation period.
- *Resilience*: Resilience is measured by how quickly the system can recover after successive norm violations. In our experiment, it is measured by the average time it takes for the system to recover to an acceptable state after falling into a bad state. We define a system with load $<= 0.4$ as being in a bad state and recognize it as recovered if load increases to a value $>= 0.7$. The defined threshold is arbitrarily-assigned and is intended to allow the simulation to capture the slope of rebounding curves. We record all occurrences of this phenomenon and average them over the course of the entire simulation, regarding smaller values as more resilient than larger values.
- *Total Research Tasks*: The number of research tasks that are completed in addition to those that are not completed (and past deadline) by the end of the simulation.
- *Completed Research Tasks*: The number of completed research tasks by all the agents throughout the simulation.
- *Total Security Tasks*: The number of security tasks that are completed in addition to those that are not completed (and past deadline) by the end of the simulation.
- *Completed Security Tasks*: The number of completed security tasks by all the agents throughout the simulation.
- *Violations*: The total number of norm violations committed throughout the run of the simulation. Note that if an agent didn't complete a security task while it's fixing its PC after being sanctioned, it doesn't count towards a norm violation because the agent is doing security measures. Thus, it is possible that the "total security tasks" value may not be equivalent to the sum of completed security tasks and violations.
- *Sanctions*: The total number of sanctions issued during the simulation. For individual sanction (S_i), it is calculated by the total number of sanctions issued to individual violators. For group sanction (S_g), it is calculated by the total number of sanctions issued to the group, despite the number of agents in the group.

4.4 Evaluation

We averaged results of each metric over 50 simulations, each with 1000 ticks (or until all the agents leave the organization, whichever the earliest), under three agent population sizes (100, 500, and 1,000). Tables 1–6 illustrate the results of our treatments with metrics as defined above.

4.4.1 Network Sizes

We ran our simulation for three different network sizes: small (100 agents), medium (500 agents), and large (1,000 agents). Results demonstrate that for variable observability (from 0% to 100%), the size of the network has no quantifiable affect on system research motivation, system security compliance, or system load.

Table 1: Individual Sanction Results for 100 Agents over 50 Simulations
O – Observability, M – System research motivation, C – System security compliance, S_i – Sanctions in individual sanction, S_g – Sanctions in group sanction

O	M	C	Load	Resilience	Research Tasks		Security Tasks			
					Total	Completed	Total	Completed	Violations	S_i
0%*	0.68	0	0	1.73	3216	1330	2087	1137	950	0
20%*	0.63	0.28	0.43	1.73	3804	1767	2814	1829	974	194
40%	0.44	0.45	0.71	2.42	11487	8524	14243	9656	4587	1833
60%	0.43	0.56	0.72	2.33	11489	8619	14268	10752	3515	2111
80%	0.43	0.64	0.72	2.65	11497	8675	14271	11365	2907	2331
100%	0.42	0.7	0.73	2.51	11498	8716	14274	11741	2533	2533

* – Simulations stop at ticks between 300 – 450, # – A small portion of simulations stop before 1000 ticks

Table 2: Group Sanction Results for 100 Agents over 50 Simulations

O	M	C	Load	Resilience	Research Tasks		Security Tasks			
					Total	Completed	Total	Completed	Violations	S_g
0%*	0.68	0	0	1.73	3207	1324	2074	1136	938	0
20%$^{\#}$	0.44	0.74	0.7	2.5	10510	7675	12603	10193	2400	26
40%	0.43	1	0.75	2.12	11501	8698	14282	12487	1795	56
60%	0.44	1	0.76	1.93	11490	8584	14281	12325	1956	86
80%	0.46	1	0.76	1.79	11493	8470	14279	12107	2172	115
100%	0.48	1	0.77	1.68	11504	8355	14277	11908	2369	142

Table 3: Individual Sanction Results for 500 Agents over 50 Simulations

O	M	C	Load	Resilience	Research Tasks		Security Tasks			
					Total	Completed	Total	Completed	Violations	S_i
0%*	0.71	0	0	1.52	16077	6651	10438	5677	4761	0
20%*	0.65	0.28	0.33	1.52	19009	8815	14021	9133	4835	969
40%	0.44	0.45	0.71	1.52	57454	42636	71227	48317	22911	9162
60%	0.43	0.56	0.72	1.52	57476	43123	71342	53773	17568	10545
80%	0.42	0.64	0.73	1.52	57471	43377	71362	56831	14531	11628
100%	0.42	0.7	0.73	1.55	57481	43552	71379	58746	12633	12633

Table 4: Group Sanction Results for 500 Agents over 50 Simulations

O	M	C	Load	Resilience	Research Tasks		Security Tasks			
					Total	Completed	Total	Completed	Violations	S_g
0%*	0.71	0	0	1.52	16068	6642	10423	5683	4740	0
20%$^{\#}$	0.44	0.73	0.71	2.27	52151	37814	62148	50020	12108	25
40%	0.43	0.99	0.76	1.95	57480	43474	71405	62416	8989	56
60%	0.44	1	0.76	1.79	57484	42964	71403	61673	9730	85
80%	0.46	1	0.76	1.68	57461	42374	71392	60618	10774	113
100%	0.47	1	0.77	1.65	57466	41747	71386	59502	11884	142

Table 5: Individual Sanction Results for 1000 Agents over 50 Simulations

O	M	C	Load	Resilience	Research Tasks		Security Tasks			
					Total	Completed	Total	Completed	Violations	S_i
0%*	0.72	0	0	1.51	32125	13269	20917	11346	9571	0
20%*	0.65	0.28	0.26	1.51	37862	17514	27871	18140	9626	1924
40%	0.44	0.45	0.71	1.51	114867	85240	142467	96592	45875	18325
60%	0.43	0.56	0.72	1.51	114977	86257	142682	107523	35159	21093
80%	0.42	0.64	0.73	1.51	114968	86761	142730	113631	29099	23277
100%	0.42	0.7	0.73	1.51	114986	87125	142755	117479	25276	25276

Table 6: Group Sanction Results for 1000 Agents over 50 Simulations

O	M	C	Load	Resilience	Research Tasks		Security Tasks			
					Total	Completed	Total	Completed	Violations	S_g
0%*	0.71	0	0	1.51	32136	13277	20892	11351	9541	0
20%$^{\#}$	0.44	0.74	0.73	2.27	106019	78166	128307	104718	23531	26
40%	0.43	1	0.76	1.88	115006	86943	142823	124917	17905	56
60%	0.44	1	0.76	1.73	114953	85927	142805	123414	19391	84
80%	0.46	1	0.76	1.66	114995	84784	142785	121245	21540	114
100%	0.47	1	0.76	1.65	114947	83501	142773	119023	23750	142

For individual sanction, resilience has less meaning in our setting. In a medium or large network, there are between 0 and 1 instances that the phenomenon occurs where system load increases from the lower threshold, <= 0.4, to the higher threshold, >= 0.7. The <= 0.4 value usually occurs at the first tick of the simulation. For individual sanction in a small network, load varies more than that in the other two networks, exhibiting instances where there are 0 – 9 instances where load increases from <= 0.4 to >= 0.7. For group sanction, the number of intervals is almost equal to the number of sanctions. Interestingly, resilience for group sanction slightly decreases with increase in network size from small to medium, and stabilizes as the network grows to large.

4.4.2 Observability

For both sanction types, at lower observability agents have a high system research motivation. However, at a higher observability (> 20% for Group Sanction and > 30% for Individual Sanction), there is no significant difference in system research motivation. Also, with increase in observability, the system security compliance increases under any sanction in a network of any size since larger observability leads to more sanctions.

At lower observability (< 20% for Group Sanction and < 30% for Individual Sanction), agents tend not to complete security tasks due to less sanctions. As a result, their respective PC health values decrease to zero (PC health threshold of compromising) eventually. Despite high research motivation, agents continually fail to complete research tasks due to zero PC health. Later, agent health values of all agents drop to zero (the agent health threshold for leaving the lab) because of unfinished research tasks, and the simulation ends before 1000 ticks, as indicated by * and # in the tables.

4.4.3 Sanction Types

Group sanction leads to slightly greater system research motivation than individual sanction when observability > 40%. For smaller observability, many simulations stopped before 1000 ticks, thus it is not meaningful to compare their values to those with larger observability. Further, student agents are more willing to comply with the security policies under group sanction, with violations occurring less frequently when the sanctioner has higher observability and more frequently with lower observability. It is also evident that individual sanction leads to 6–20 times greater total number of sanctions than group sanction, which suggests that the cost of individual sanction is much higher than that of group sanction. Thus group sanction may be adopted as a cost-effective approach.

4.4.4 Tasks

In group sanction, student agents are driven to give more priority to security tasks when sanctioned, thus leading to more completed security tasks than in individual sanction. During the period that the agents are trying to fix their PCs after being sanctioned, they are behind schedule on their research tasks, and as a result, less research tasks are completed.

5. RELATED WORKS

Our interest in the costs related to sanction arose as a result of surveying literature concerning low-level representations of normative relationships. Dissatisfied with our findings, we constructed an analogy (the research lab scenario) that encapsulated the aspects of norms we were interesting in investigating. While tuning the variables of our simulation, we began to question the proper mode of sanction to impose on the agents for norm violation, which lead us back to the literature for answers. With no applicable response, we were determined to formulate our own analysis.

Previous works in observability and cost-efficiency are primarily concerned with decentralized systems (i.e., no administrating principal), where agents participate in a cooperative game (see [3, 10, 16]). Mahmoud et al. impose a game on a set of agents who are required to make binary decisions to either comply or defect when presented with a norm. Norm violations are met with punishment, a form of consequence specifically targeted at destabilizing the economic gain of the defector [16]. The agent's decisions also affects its reputation and sets precedence for how it is to be treated later in the simulation by other agents. The ability for an agent to judge another based on reputation is dictated by that agent's ability to observe its environment. Mahmoud et al. conclude greater observability yields stronger norm compliance via punishment, which motivated our interest in performing similar analyses on sanction.

Villatoro et al. evaluate the cost-effectiveness of sanction against punishment. Their simulation randomly pairs agents as players of a Prisoner's Dilemma game. The agents are, again, allowed the binary option to comply or defect, and the agent's decision is persistent in subsequent rounds of the game. Agents that comply are able to sanction others in later stages to communicate norm compliance, whereas those that defect must use punishment. Villatoro et al. conclude that sanction is a much more cost-efficient means of garnering cooperation, which lends confidence to the relevance of our study.

In contrast to the aforementioned studies, our simulation recognizes sanction as the sole means for applying consequence to norm violation. This follows from our structured lab scenario where directed normative relationships exist from an adviser to his or her graduate student and from the network administrator to the students in the lab. Further, our simulation is not decentralized, as there exists a central authority who's responsible for all sanctions.

6. CONCLUSIONS AND FUTURE WORK

Using ENGMAS, our exploration of system resilience and liveness through variable observability and sanction type has yielded some interesting conjectures. For example, if the sanctioner has low observability, agents are less "controlled" and tend to violate security norms more frequently. As a result, their respective PCs enter a critical state, decreasing their utility in performing research tasks until they are let go from the organization. Student agents perform better under group sanction than under individual sanction in terms of total norm violations. Considering the time-cost associated with issuing sanctions, these results may suggest that it is more cost-effective to govern with group sanction, as the sanctioner may maintain security compliance with far fewer sanctions.

In other future work, we plan to further develop our scenario and its accompanying simulation in order to conduct more in-depth research on norm-governance in multiagent systems and we anticipate more variance on resilience. Incorporation of multidisciplinary concepts from fields including anthropology, sociology, and psychology, among others, will help us to refine the behavior of our agents and increase the potential for more evaluative studies on norms and their properties. We further anticipate building a realistic model of agents with rational choices based on rewards, sanctions, and utilities, and their capabilities of observation and communication, and including cost of applying sanctions and the norm sanctioning capabilities of the governing principal to better portray the reality of implementing such policies.

Acknowledgments

We gratefully acknowledge National Security Agency for support via the Science of Security Lablet at North Carolina State University. We would like to thank Jon Stallings for helpful comments on a previous version.

REFERENCES

[1] R. M. Adams. Strategies of maximization, stability, and resilience in Mesopotamian society, settlement, and agriculture. *Proceedings of the American Philosophical Society*, 122(5):329–335, 1978.

[2] B. Alpern and F. B. Schneider. Defining liveness. *Information Processing Letters*, 21(4):181–185, 1985.

[3] R. Axelrod. An evolutionary approach to norms. *American Political Science Review*, 80(04):1095–1111, 1986.

[4] F. Berkes and N. Turner. Knowledge, learning and the evolution of conservation practice for social-ecological system resilience. *Human Ecology*, 34(4):479–494, 2006.

[5] M. Bishop. What is computer security? *IEEE Security Privacy*, 1(1):67–69, Jan 2003.

[6] R. L. Boring. Reconciling resilience with reliability: The complementary nature of resilience engineering and human reliability analysis. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 53, pages 1589–1593. Sage Publications, 2009.

[7] G. Dhillon and J. Backhouse. Technical opinion: Information system security management in the new millennium. *Communications of the ACM*, 43(7):125–128, July 2000.

[8] D. D. Heckathorn. Collective sanctions and compliance norms: A formal theory of group-mediated social control. *American Sociological Review*, 55(3):366–384, 1990.

[9] L. Lamport. Proving the correctness of multiprocess programs. *IEEE Transactions on Software Engineering*, SE-3(2):125–143, March 1977.

[10] S. Mahmoud, D. Villatoro, J. Keppens, and M. Luck. Optimised reputation-based adaptive punishment for limited observability. In *Self-Adaptive and Self-Organizing Systems (SASO), 2012 IEEE Sixth International Conference on*, pages 129–138, Sept 2012.

[11] P. Noriega, A. K. Chopra, N. Fornara, H. L. Cardoso, and M. P. Singh. Regulated MAS: Social Perspective. In G. Andrighetto, G. Governatori, P. Noriega, and L. W. N. van der Torre, editors, *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*, pages 93–133. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013.

[12] R. Plummer and D. Armitage. A resilience-based framework for evaluating adaptive co-management: Linking ecology, economics and society in a complex world. *Ecological Economics*, 61(1):62–74, 2007.

[13] T. B. Sheridan. Risk, human error, and system resilience: Fundamental ideas. (cover story). *Human Factors*, 50(3):418–426, 2008.

[14] M. P. Singh. Norms as a basis for governing sociotechnical systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(1):21:1–21:23, Dec. 2013.

[15] W. Vasconcelos, M. Kollingbaum, and T. Norman. Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 19(2):124–152, 2009.

[16] D. Villatoro, G. Andrighetto, J. Sabater-Mir, and R. Conte. Dynamic sanctioning for robust and cost-efficient norm compliance. In *International Joint Conferences on Artificial Intelligence*, volume 11, pages 414–419, 2011.

APPENDIX

Table 7: Variables with Normal Distribution

Variable	Value
μ of Preference	0.5
σ of Preference	0.3
Upper Limit of Preference	0.8
Lower Limit of Preference	0.4
μ of Research Motivation	0.7
σ of Research Motivation	0.15
μ of Security Compliance	0.7
σ of Security Compliance	0.15
μ of Research Task Duration	5
σ of Research Task Duration	3
Lower Limit of Research Task Duration	1

Table 8: Experiment Parameters

Experiment Parameter	Value
Coefficient c for Research Task Duration	1.3
Naturally Decrease Rate of PC Health	0.1
Increase Rate of Agent Health	0.25
Decrease Rate of Agent Health	0.25
Increase Rate of PC Health	0.25
Decrease Rate of PC Health	0.25
Increase Rate of Research Motivation	0.25

Towards Implicit Contextual Integrity

(Position Paper)

Natalia Criado
School of Computer Science
Liverpool John Moores University
Liverpool, UK
n.criado@ljmu.ac.uk

Jose M. Such
School of Computing and Communications
Lancaster University
Lancaster, UK
j.such@lancaster.ac.uk

ABSTRACT

Many real incidents demonstrate that users of Online Social Networks need mechanisms that help them manage their interactions by increasing the awareness of the different contexts that coexist in Online Social Networks and preventing users from exchanging inappropriate information in those contexts or disseminating sensitive information from some contexts to others. Contextual integrity is a privacy theory that expresses the appropriateness of information sharing based on the contexts in which this information is to be shared. Computational models of Contextual Integrity assume the existence of well-defined contexts, in which individuals enact pre-defined roles and information sharing is governed by an explicit set of norms. However, contexts in Online Social Networks are known to be implicit, unknown a priori and ever changing; users' relationships are constantly evolving; and the norms for information sharing are implicit. This makes current Contextual Integrity models not suitable for Online Social Networks. This position paper highlights the limitations of current research to tackle the problem of exchanging inappropriate information and undesired dissemination of information and outlines the desiderata for a new vision that we call *Implicit* Contextual Integrity.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Security

Keywords

Contextual Integrity, Privacy, Online Social Networks, Norms

1. INTRODUCTION

Online Social Networks (OSNs) have been a source of privacy concerns and issues since their early days [19]. These privacy concerns have increased along the past decade due to many real privacy incidents being echoed in the media and users being more aware of potential privacy issues [11, 35]. Yet there is a lack of effective privacy controls that allow users to satisfactorily manage their privacy in OSNs [34]. In particular, the exchange of inappropriate information and the undesired dissemination of sensitive information across OSNs are very common and represent one of the major concerns for users. These inappropriate exchanges and undesired disseminations have not only caused serious privacy incidents — e.g., users have lost their jobs [28], have been outed and faced threats to sever family ties [17], have ended their marriages [32], etc. — but also facilitated other activities such as social phishing [21], identity theft [3], cyberstalking [25], and cyberbullying [30].

Some voices argue that this is mainly due to the fact that users are no longer able to share information differently for different contexts or spheres of life (friends, work, etc.) in the cyber world, as they would usually do in the physical world [40]. There are many examples in which this is of crucial importance: photos that depict users in embarrassing situations, indecorous comments, events that reveal some political affiliations, etc. In all these examples, the specific context determines whether or not the exchange of information is appropriate — e.g., one may be willing to share her political affiliations with friends but not with workmates.

Contextual integrity [27] is a modern privacy theory that expresses the appropriateness of information sharing based on the contexts in which this information is to be shared. In particular, contexts are defined considering a set of individuals playing particular roles and a number of norms that govern information sharing among them. Contextual integrity is said to be maintained — meaning that there are no privacy breaches — whenever these norms for information sharing are upheld. Norms for information sharing have two main purposes: (i) determine what information is appropriate to mention in a particular context, and (ii) dictate what information can be transmitted from one party to another or others according to the roles enacted by these parties within and across different contexts.

Computational models of contextual integrity have been recently proposed in the related literature [1, 24]. Following contextual integrity theory, they assume the existence of well-defined contexts, in which individuals enact pre-defined roles and information sharing is governed by an explicit set of norms. However, contexts in OSNs are "implicit, ever changing and not a priori-known" [12]. In particular, norms for information sharing are known to be *implicit* in OSNs [29, 39], i.e., they define the behaviour that is consistent with the most common behaviour. Moreover, roles are dynamic and may not be known a priori — i.e., relationships among individuals in OSNs are constantly evolving [5]. All of these reasons make explicit contextual integrity and the computational models based on it not suitable for OSNs.

In this paper, we argue that a new computational paradigm for Contextual Integrity is needed such that it supports implicit norms for information sharing and contexts as well as dynamic and not-a-priori-known roles. We call such an approach *Implicit* Contextual Integrity.

2. PROBLEM STATEMENT

This paper tackles the following two privacy threats in OSNs:

2.1 Inappropriate Information Exchange

Each context has its own appropriateness norms that determine which information can be mentioned inside each context. For example, one may not mention her political views in a work context, but she may do so in a family context [40]. New models of implicit contextual integrity can use the information that users of OSNs exchange with other users in one context (or community) to infer the appropriateness norms of this context. Specifically, the information that is frequently mentioned by the members of a context can be considered as appropriate whereas information that is never or rarely mentioned can be considered as inappropriate. For instance, if most people do not mention their political views at work, it could be inferred this is not an appropriate topic to exchange in a work context.

Besides the appropriateness norms of each context, there are situations in which people decide to exchange information that may be seen as inappropriate. One of the main reasons that explain this is the creation and reciprocation of close relationships [18]. Indeed, there are empirical studies that demonstrate the fact that reciprocated communication is the dominant form of interaction in OSNs [8]. Accordingly, models of implicit contextual integrity can take into account the appropriateness of the information that has been exchanged with each user to determine when inappropriate information is being exchanged to reciprocate a close friend or to create a close relationship.

2.2 Undesired Information Dissemination

Dissemination occurs when information disclosed in one context travels to another context. That is, dissemination is inter-context disclosure while exchange (as stated above) is intra-context disclosure. Obviously, if the information to be disclosed is already known in the contexts were it may be disclosed, then the disclosure of this information in these contexts cannot entail any new privacy risk. However, disseminations may potentially be undesired and hazardous when they entail the disclosure of sensitive information that was previously unknown in a context [33]. Indeed, studies on regrets associated to users' posts on OSNs highlight the fact that revealing secrets of others is one of the main sources of regret [40]. For instance, there was a recent case in which the sexuality of a person was leaked from her friends context to her family context where her sexuality was previously unknown, causing her being outed and facing threats to sever family ties [17].

A first-line defence against undesired disseminations may be avoiding sharing sensitive information in contexts in which there are people that could disseminate the information to other contexts in which this information is previously unknown. Whether these people decide to disseminate the information or not may depend on the relationship they have to others. That is, people usually have confidence relationships with others with whom they decide to share sensitive information expecting them to keep it secret. One can share some of her deepest secrets with her husband but this may not mean her husband would disseminate this information to other contexts. Thus, models of implicit contextual integrity could take into account the knowledge of the information that has been exchanged with each user to determine when sensitive information is being exchanged to reciprocate a trusted friend or to create/maintain trust relationships.

3. LIMITATIONS OF RELATED WORK

In this section we discuss why current approaches are not enough to deal with Inappropriate Information Exchange and Undesired Information Dissemination in OSNs.

3.1 Contextual Integrity Modelling and Reasoning

Previous work on computational models of contextual integrity proposed mechanisms for modelling and reasoning about contextual integrity principles. For example, Barth et al. [1] formalized some aspects of contextual integrity assuming that there is a set of explicitly defined norms that determine what is permitted and forbidden, that the interactions take place in well-known contexts, and that interaction participants play a specific role in each interaction. In a more recent work, Krupa et al. [24] proposed a framework to enforce norms for information sharing in an electronic institution where norms, contexts and roles are explicitly defined. While these approaches seem appropriate for the kind of domains described in [1] and [24], in OSNs there are not well-known contexts, there is not an explicit definition of the roles played by users and the exchange of information is governed by implicit norms for information sharing. Note that these implicit norms for information sharing define the behaviour that is consistent with the most common behaviour. In contrast, explicit norms for information sharing define behaviour that is normative (i.e., moral).

3.2 Access Control Models for OSNs

The suitability of traditional access control models such as role-based access control (RBAC) for OSNs has been recently challenged on the basis that they cannot capture the inherent social nature of OSNs, such as social relationships and distance among users. To address this limitation, there is a new paradigm that precisely emphasises the social aspects of OSNs. Access control models in this new paradigm are commonly referred to as Relationship-based Access Control (ReBAC) [14, 16, 15, 4, 7, 6, 9]. ReBAC models utilise a variety of features or aspects to characterise users' relationships and define access control decisions based on them. While ReBAC models represent a better framework than other traditional access control approaches to develop tools for defining and enforcing access control policies in OSNs, access control on its own is unlikely to be the complete and definitive solution for an appropriate privacy management in OSNs, as users need awareness about access control decisions to fully understand the consequences of their access control policies [22, 26]. For instance, access control models are known to fail to prevent unintended disclosures [2].

3.3 Disclosure Decision-Making Mechanisms

In the related literature, the use of software endowed with disclosure decision-making mechanisms is not new. For example, several authors [37, 23] proposed mechanisms for computing the privacy-benefit trade-off of information disclosures in online interactions. The aim is to only disclose information when this trade-off renders appropriate results, i.e., where the utility of a particular disclosure is worth the privacy risks/consequences involved by performing the disclosure. However, these mechanisms have difficulties to deal with scenarios where the direct benefit of disclosing a piece of information is a priori unknown or difficult to express in economic terms, such as OSNs, in which disclosures are mostly driven by social factors [20]. In a more recent work, Such et al. [36] proposed a mechanism for entailing agents with capabilities to select the personal attributes of their users to be disclosed to other agents during interactions considering the increase on intimacy and privacy loss a disclosure may cause. However, this mechanism does not consider that the appropriateness of disclosures may vary from context to context, nor does it consider information disseminations.

3.4 Norm Learning

Norm learning [13] is the process of learning how to behave in a specific situation. In case of OSNs, norms for information sharing are implicit (i.e., there is not an explicit definition of what is sensitive or inappropriate), and supervised machine learning algorithms cannot be used to infer norms for information sharing. In the existing literature, *social learning* [10] of norms is defined as the process of inferring implicit social norms concurrently over repeated interactions with members of the social network. In most of the proposals on social learning, norms are inferred by analysing the outcomes of interactions and normative decisions in terms of utility [31]. As previously mentioned, in OSNs the benefit of exchanging information may be difficult to be determined in economic terms. In other proposals, norms are inferred by analysing explicit normative signals such as punishments, sanctions and rewards [38]. These approaches cannot be used in OSNs since implicit norms for information sharing are product of informal social control that is rarely stated explicitly (e.g., sanctions) to unfriendly individuals. Other approaches [13] use imitation as a mechanism for learning social norms. In these proposals, the norms are inferred from the public behaviour exhibited by the majority of the members of the social network (or the majority of the members within an observation radius). A main drawback of imitation approaches is that all members are equally considered; i.e., they do not consider the existence of different social contexts with different social norms and the fact that users engage in relationships of different nature and strength. These unsupervised machine learning approaches are unsuitable to be applied to ONS.

4. IMPLICIT CONTEXTUAL INTEGRITY

We propose that a new computational model of *implicit* Contextual Integrity for OSNs should be built. To be applicable in mainstream OSN infrastructures, this model should only utilise the information that is currently available to users of OSNs and their applications —e.g., the tweets posted by users the following relationships, etc.

Our vision is to include such kind of model in what we would call *Information Assistant Agents* (IAAs), which are agents that act as proxies to access the OSN. IAAs should be capable of learning contexts and their associated norms for information sharing even if these are implicit or unknown a priori with the aim of helping users to avoid inappropriate information exchanges and undesired information disseminations. In particular, each IAA monitors the information exchanges of its user and based on this it infers: (i) the different contexts in which information sharing is to happen; (ii) the relationships between its user and the individuals in each context; and (iii) the norms for information sharing of each context. If IAAs detect a potential violation of the norms for information sharing, they should alert their users, who should have the last word on whether sharing the information or not.

REFERENCES

[1] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum. Privacy and contextual integrity: Framework and applications. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 184 – 198, 2006.

[2] M. S. Bernstein, E. Bakshy, M. Burke, and B. Karrer. Quantifying the invisible audience in social networks. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 21–30, 2013.

[3] L. Bilge, T. Strufe, D. Balzarotti, and E. Kirda. All your contacts are belong to us: automated identity theft attacks on social networks. In *Proceedings of the International Conference on World Wide Web*, pages 551–560, 2009.

[4] G. Bruns, P. W. Fong, I. Siahaan, and M. Huth. Relationship-based access control: Its expression and enforcement through hybrid logic. In *Proceedings of the ACM Conference on Data and Application Security and Privacy*, pages 117–124. ACM, 2012.

[5] M. Burke and R. E. Kraut. Growing closer on facebook: changes in tie strength through social network site use. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 4187–4196, 2014.

[6] B. Carminati, E. Ferrari, and A. Perego. Rule-based access control for social networks. In R. Meersman, Z. Tari, and P. Herrero, editors, *On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops*, volume 4278 of *Lecture Notes in Computer Science*, pages 1734–1744. Springer Berlin Heidelberg, 2006.

[7] B. Carminati, E. Ferrari, and A. Perego. Enforcing access control in web-based social networks. *ACM Transactions on Information and System Security*, 13(1):6:1–6:38, Nov. 2009.

[8] J. Cheng, D. M. Romero, B. Meeder, and J. M. Kleinberg. Predicting reciprocity in social networks. In *Privacy, Security, Risk and Trust, IEEE International Conference on Social Computing*, pages 49–56, 2011.

[9] Y. Cheng, J. Park, and R. Sandhu. A user-to-user relationship-based access control model for online social networks. In N. Cuppens-Boulahia, F. Cuppens, and J. Garcia-Alfaro, editors, *Data and Applications Security and Privacy XXVI*, volume 7371 of *Lecture Notes in Computer Science*, pages 8–24. Springer Berlin Heidelberg, 2012.

[10] N. Criado, E. Argente, and V. J. Botti. Open issues for normative multi-agent systems. *AI Communications*, 24(3):233–264, 2011.

[11] B. Danah and E. Hargittai. Facebook privacy settings: Who cares? *First Monday*, 15(8), 2010.

[12] G. Danezis. Inferring privacy policies for social networking services. In *Proceedings of the ACM workshop on Security and artificial intelligence*, pages 5–10, 2009.

[13] J. M. Epstein. Learning to be thoughtless: Social norms and individual computation. *Computational Economics*, 18(1):9–24, 2001.

[14] P. Fong, M. Anwar, and Z. Zhao. A privacy preservation model for facebook-style social network systems. In M. Backes and P. Ning, editors, *Computer Security âĂŞ ESORICS 2009*, volume 5789 of *Lecture Notes in Computer Science*, pages 303–320. Springer Berlin Heidelberg, 2009.

[15] P. W. Fong. Preventing sybil attacks by privilege attenuation: A design principle for social network systems. In *IEEE Symposium on Security and Privacy*, pages 263–278, 2011.

[16] P. W. Fong. Relationship-based access control: Protection model and policy language. In *Proceedings of the ACM Conference on Data and Application Security and Privacy*, pages 191–202, 2011.

[17] G. A. Fowler. When the most personal secrets get outed on facebook. `http://online.wsj.com/articles/SB10000872396390444165804578008740578200224`, Accessed: Nov, 2014.

[18] K. Greene, V. J. Derlega, and A. Mathews. Self-disclosure in personal relationships. *The Cambridge handbook of personal relationships*, pages 409–427, 2006.

[19] R. Gross and A. Acquisti. Information revelation and privacy in online social networks. In *Proceedings of the ACM workshop on Privacy in the electronic society*, pages 71–80, 2005.

[20] D. J. Houghton and A. N. Joinson. Privacy, social network sites, and social relations. *Journal of Technology in Human Services*, 28(1-2):74–94, 2010.

[21] T. N. Jagatic, N. A. Johnson, M. Jakobsson, and F. Menczer. Social phishing. *Communications of the ACM*, 50(10):94–100, 2007.

[22] L. Kagal and H. Abelson. Access control is an inadequate framework for privacy protection. In *W3C Privacy Workshop*, pages 1–6, 2010.

[23] A. Krause and E. Horvitz. A utility-theoretic approach to privacy and personalization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 8, pages 1181–1188, 2008.

[24] Y. Krupa and L. Vercouter. Handling privacy as contextuxal integrity in decentralized virtual communities: The privacias framework. *Web Intelligence and Agent Systems*, 10(1):105–116, 2012.

[25] A. Lyndon, J. Bonds-Raacke, and A. D. Cratty. College students' facebook stalking of ex-partners. *Cyberpsychology, Behavior, and Social Networking*, 14(12):711–716, 2011.

[26] M. Mondal, P. Druschel, K. P. Gummadi, and A. Mislove. Beyond Access Control: Managing Online Privacy via Exposure. In *Proceedings of the Workshop on Useable Security*, pages 1–6, 2014.

[27] H. Nissenbaum. Privacy as contextual integrity. *Washington Law Review*, 79(1):119–158, 2004.

[28] G. H. Pike. Fired over facebook. *Information Today*, 28(4):26–26, 2011.

[29] K. Raynes-Goldie and D. Fono. Hyperfriends and beyond: Friendship and social norms on livejournal. *Internet Research Annual*, 4:8, 2006.

[30] M. C. Ruedy. Repercussions of a myspace teen suicide: Should anti-cyberbullying laws be created. *North Carolina Journal of Law & Technology*, 9:323–346, 2007.

[31] S. Sen and S. Airiau. Emergence of norms through social learning. In *Proceedings of the International Joint Conference on Artifical Intelligence*, pages 1507–1512, 2007.

[32] J. Stevens. The facebook divorces: Social network site is cited in 'a third of splits'. `http://www.dailymail.co.uk/femail/article-2080398/Facebook-cited-THIRD-divorces.html`, Accessed: Nov, 2014.

[33] L. J. Strahilevitz. A social networks theory of privacy. In *American Law & Economics Association Annual Meetings*, pages 919–988, 2005.

[34] K. Strater and H. R. Lipford. Strategies and struggles with privacy in an online social networking community. In *Proceedings of the HCI Group Annual Conference on People and Computers: Culture, Creativity, Interaction*, pages 111–119, 2008.

[35] F. Stutzman, R. Gross, and A. Acquisti. Silent listeners: The evolution of privacy and disclosure on facebook. *Journal of Privacy and Confidentiality*, 4(2), 2013.

[36] J. M. Such, A. Espinosa, A. Garcia-Fornes, and C. Sierra. Self-disclosure decision making based on intimacy and privacy. *Information Sciences*, 211(0):93 – 111, 2012.

[37] S. van Otterloo. The value of privacy: optimal strategies for privacy minded agents. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1015–1022, 2005.

[38] D. Villatoro, G. Andrighetto, J. Sabater-Mir, and R. Conte. Dynamic sanctioning for robust and cost-efficient norm compliance. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 414–419, 2011.

[39] M. Vorvoreanu. Perceptions of corporations on facebook: An analysis of facebook social norms. *Journal of New Communications Research*, 4(1):67–86, 2009.

[40] Y. Wang, G. Norcie, S. Komanduri, A. Acquisti, P. G. Leon, and L. F. Cranor. I regretted the minute i pressed share: A qualitative study of regrets on facebook. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, page 10, 2011.

Towards scalable network host simulation

Jan Stiborek
Czech Technical University in Prague, Czech Republic
CISCO Systems, Inc.
jastibor@cisco.com

Martin Rehák
CISCO Systems, Inc.

Tomáš Pevný
Czech Technical University in Prague, Czech Republic

ABSTRACT

Anomaly detection techniques in network security face significant challenges on configuration and evaluation, as collecting data for accurate analysis is difficult or nearly impossible. One viable approach is to avoid live data collection and replace if by the agent-based simulation of the network traffic with models of user's behavior. In this paper we propose three approaches differing by the level of detail with which user behavior is modeled. They are well suited for generating NetFlow/IPFIX data that can be used for evaluation and optimal configuration of anomaly detection techniques. First two techniques use simple statistical model that is easy to implement and does not require large amount of training data. The third leverages sophisticated model of the user's behavior covering different aspects of the network traffic not captured by the simpler models. In experimental evaluation it is demonstrated that the complex model generates data indistinguishable for current state-of-the-art anomaly detection methods from the real-world samples, which makes it well-suited for their evaluation and configuration.

Categories and Subject Descriptors

[**Network**]: Network simulations; [**Security and privacy**]: Intrusion detection systems

General Terms

Security,Algorithms

Keywords

Net-Flow simulation,anomaly detection,evaluation

1. INTRODUCTION

This paper presents a user's behavior models for agent-based simulation network traffic that can be used for evaluation of an intrusion detection system (IDS) composed of NetFlow-based anomaly detectors [1, 2]. The main reason why researchers focus on the NetFlow data is that it captures only high level statistics which have been shown sufficient [3] for detecting threats and thus allows to process data from high speed backbone networks which cannot be achieved with other techniques (e.g. deep packet inspection).

The main and prevalent problem is the lack of ground truth data which is due to: (i) unrealistic properties of the ground truth generated in the closed lab without real-world background traffic; (ii) huge volumes of data to label in real environments; and (iii) varying characteristics between different environments making model transfer difficult.

We show that a viable solution to the above problems is to simulate the network traffic. Existing approaches can be divided into following areas: (i) context-free packet generators that correctly capture properties of individual packets but not the high level properties of user's behavior [4–8]; (ii) testbed systems that can generate the network traffic on packet level but are extremely difficult to setup [9]; (iii) lightweight simulator generating only the NetFlow data [10–13].

Our work belongs to the last category of lightweight simulators. Our goal is a realistic simulation of the background traffic (NetFlow records) with correct high level properties, for which we propose three techniques. The first two uses a simple statistical model to generate training data. They are both easy to implement but do not capture sophisticated aspects of user's behavior, such as time variance of the user's behavior, dependency between inter-flow features, etc. The third one, which we advocate, addresses these deficiencies and based on our results is able to mimic the user's traffic in a way that even a combination of state-of-the-art detection algorithms is not able to distinguish.

2. RELATED WORK

Network simulation provides viable approach for ground truth generation as discussed in [14,15]. Authors argue that static data sets and manual labeling suffer serious problems that network simulation can overcome (manual labeling does not scale, bias in the labeled data, privacy issues with sharing of labeled data, etc.).

First group of traffic generators are context-free packet generators that generate full packet captures, such as NS-2 and NS-3 simulators [4, 5], OMNET++ [6] or NeSSi [7] and its ancestor NeSSi2 [8]. However, these tools were primary designed to test low level algorithms (e.g. routing algorithms) and do not model the high level statistics necessary for evaluation of an IDS system. Moreover, these tools require precise configuration which drastically increases the cost of their deployment.

Next option for generating ground truth data is the testbed systems that emulate the behavior of the network. Such solution was proposed in project LARIAT [9] where authors used virtual machines with service that emulates user's behavior. However, as authors argued, LARIAT requires careful tuning which lowers the chance for practical deployment (authors claim that it takes approximately four months to setup LARIAT for evaluation of new IDS system). Our work is inspired by the high level design of the LARIAT system. However, the key difference between our work and LARIAT system is that we simulate the user's behavior with statistical models trained from the data and LARIAT uses emulation which requires manual setup.

Another approach is the flow-level simulation that models particular type of behavior, e.g. SSH brute-force attack, which is mixed with background traffic and used for evaluating an IDS system. Such approach is proposed in [10,11].

The most advanced and the most challenging approach is to model set of users or the whole network and generate the complete dataset (without the need to mix with background traffic). Such approach is well suited for evaluation of an IDS system because the data (if generated correctly) mimic the behavior of the whole network, can be arbitrarily tuned (duration, volume of traffic, number of users, etc.) and can be shared between researches without any privacy concerns. In [13] authors propose host-agent based simulator where single class of network behavior is represented by an autonomous agent (trained from sample traces or malicious traffic model). The traffic is then generated from interaction between agents. In [12] authors propose approach based on modified version of *Traffic Dispersion Graphs* which define the connectivity patterns for given service. The port-based TDGs augmented with additional statistics such as distribution of packets, bytes or duration serve as a model that is able to generate the traffic traces for the whole network. Our solutions use similar statistics to describe the communication between single user and requested service but the connectivity pattern is modeled by probability distributions rather than TDGs. It is designed to precisely model the behavior of single user, not the whole network. However, we can couple together multiple instances of models with different training data and simulate the behavior of the whole network.

3. BASIC MODELS

The design of a simulation model needs to address the common trade-off between complexity and performance. Before introducing our key model proposal in the next section we first discuss two simpler models. We show that simplifying assumptions about NetFlow traffic allow for models of low complexity. At the same time we will show where simplified models fail. The identified flaws then inspire the definition of the improved model presented in the next section.

3.1 Random sampling

The simplest approach to simulation of behavior of a single user is to generate standard NetFlow fields (as listed in Table 1) independently inspired by technique proposed in [16] and technological solutions such as BreakingPoint[1]. Such approach does not take into account any properties of the NetFlow (e.g. distribution of bytes, distribution of source ports, etc.), relation between fields of the NetFlow (e.g. bytes/packet ratio) or relations between individual NetFlows (e.g. request/responses relations). The only condition that has to be satisfied is the validity of NetFlow, i.e. all fields have to be in their allowed ranges and the following condition must be met

$$0 < \text{number of bytes} \leq \text{number of packets} \times 65535. \quad (1)$$

Note that the only parameter of this algorithm that has to be specified in advance is the IP address of the simulated user.

Table 1: List of NetFlow fields.

Field name	Description
Starting time	Time stamp of the first packet of the flow
Duration	Length of the flow
Protocol	TCP, UDP, ICMP, etc.
Source IP	IP address of the source
Source port	
Destination IP	IP address of the target
Destination port	
Bytes	Number of bytes transferred in the flow
Packets	Number of packets transferred in the flow
TCP flags	Not used for non-TCP protocols

The random sampling algorithm generates all but two individual NetFlow features randomly with respect to the condition of validity of the generated NetFlow. The two exceptions are the thinking time and source and destination IP addresses. Instead of generating the starting time directly we generate the user's thinking time—time delay between two consequential NetFlows. The new starting time is then computed as sum last starting time and the current thinking time. This approach allows us to generate infinite stream of NetFlows. Similarly to the thinking time, the source and destination IP addresses are not generated directly. Instead, we randomly choose whether given flow is request or response. If the flow is generated as request the source IP field is set the value of user's IP address and the destination IP is chosen randomly. In the case of response it is vice versa.

The main benefit of this algorithm is its independence on any training data or manual tuning, because the only parameter that has to be set in advance is the user's IP address. However at the same time, the complete randomness is the main disadvantage because it can generate completely unrealistic data. Therefore, we use this approach only for syntax testing and as a baseline for the comparison to more sophisticated methods.

3.2 Sampling with independent intra-flow relations—marginal model

The marginal model provides different approach to NetFlow simulation. It uses training data of a single user in order to train the statistical model of individual NetFlow features (e.g. distribution of bytes or distribution of user's thinking times, etc). Unlike the random sampling discussed above, the marginal model considers NetFlows in request/response pairs[2]. Therefore it is able to partially model inter-

[1]http://www.ixiacom.com/breakingpoint

[2]The model focus on the modeling of user's behavior and thus we consider outgoing flow as *request* and incoming flow

Table 2: List of NetFlow features to be modeled in order to create NetFlow data that correctly reflect requests and responses. Note that TCP flags for request and response are empty for all non-TCP NetFlows.

Feature name	Description
Client's thinking time	Time difference between two consequential client's requests
Client port	Source port of the request
Request bytes	Number of bytes in request
Request packets	Number of packets in request
Request protocol	TCP, UDP, ICMP, etc.
Request flags	TCP flags in request
Request length	Duration of request
Server thinking time	Time difference between request and response
Server IP	IP address of server
Server port	Port number of service used by user
Response bytes	Number of bytes in response
Response packets	Number of packets in response
Response length	Duration of response
Response flags	TCP flags in response
Has response	Is there corresponding response to the request?

flow relations (the relation between individual flows) as well because it captures request/response relations but not the sequential character of the user's behavior. For example, it is able to model the HTTP request/response pairs but not the download of the whole Google home page. Next, this model assumes that the modeled features are independent and thus it does not take into the account intra-flow relations (e.g. bytes/packets ratio, etc.). The full list of modeled features is listed in Table 2. This assumption is a limitation that affects the variance of the internal model and can cause serious sampling artifacts (see Figure 1).

Marginal models are created as follows. The first step of the sampling algorithm is the preprocessing of the training data during which we pair the requests with the corresponding responses. We assume that the user behaves only as a client and thus every NetFlow with user's IP address as source IP is considered as a request. The corresponding response is matched as NetFlow with following properties:

$$\begin{aligned}\text{source IP}_{\text{response}} &= \text{destination IP}_{\text{request}},\\ \text{source port}_{\text{response}} &= \text{destination port}_{\text{request}},\\ \text{destination IP}_{\text{response}} &= \text{user's IP},\\ \text{destination port}_{\text{response}} &= \text{source port}_{\text{response}},\\ \text{protocol}_{\text{response}} &= \text{protocol}_{\text{request}}. \qquad (2)\end{aligned}$$

Note that there is a maximal delay τ between request and response in order to avoid incorrectly paired flows. In current settings the τ is set to 2 seconds.

After the data preprocessing the model estimates the distributions of all individual features using non-parametric estimates (histogram for continuous features or relative frequencies for categorical features).

Once model training is finished, request/response pairs

as *response*.

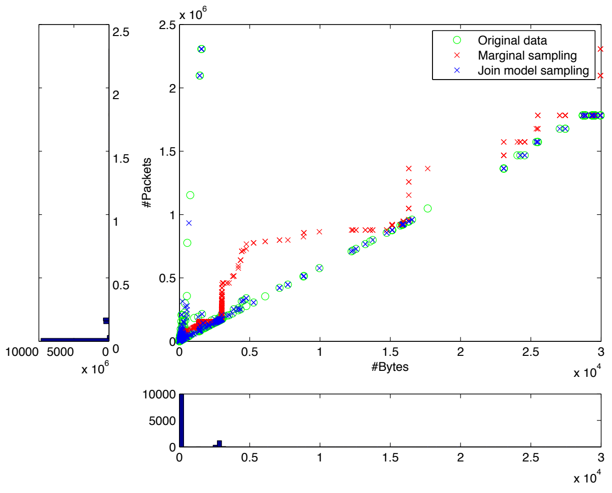


Figure 1: Artifacts of marginal sampling in real-life example. This figure shows that marginal sampling (red crosses) is not able to mimic the data correctly and thus creates serious sampling artifacts—it creates data that did not appear in the original data—whereas the joint model sampling that respects the dependency between features generates the data correctly.

are sampled as follows. At first we randomly choose whether there will be a response or it will be only request (the feature *Has response*). Next, values of individual fields are sampled from distributions of corresponding features estimated from the training data. Note that starting times of the request and response are not generated in the same manner. The starting time of the request is generated as sum of the last starting time and the user's thinking time (estimated from the training data) and the response starting time is computed as sum of starting time of the request and thinking time of the server (again, estimated from the training data). This approach is similar to the Random sampling discussed in Section 3.1. The source address in the request/destination IP in the response is set to the user's IP address (parameter of the algorithm) and the destination address in the request/source address in the response is sampled from the distribution estimated from the training data.

4. TIME VARIANT JOIN PROBABILITY MODEL

In this section we will discuss the main contribution of this paper. In previous sections we have described the simulation techniques that uses simple statistical model and thus miss more complicated aspects of the user's behavior which leads to following issues:

- no intra-flow relations—single HTTP connection will not likely transfer 60GB in 2 seconds,
- no time variant restrictions—user activity differs during the night and day,
- no reflection of sequential character of the user—HTTP request precedes a DNS request

The main problem of such approaches is the inability to model *relations between individual features*. The sampling

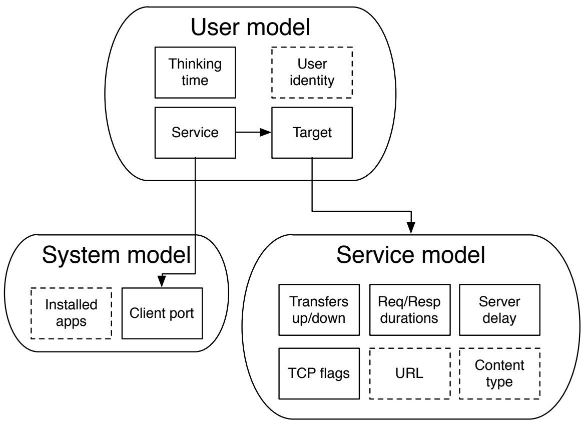


Figure 2: The schema of the proposed model. The figure shows three main components and their relations. The separation of user model and underlying system and service model is inspired by LARIAT project [9] which uses model of the user's behavior on top of the real emulated operating system and services. Note that the dashed boxes are omitted from our implementation in order to reduce the complexity of the current implementation.

with marginal model assumes that NetFlow features are independent, which if not satisfied leads to serious sampling artifacts (see Figure 1).

Next problem is caused by *changes of the user's behavior*. Usually during the night there is no network traffic generated by the user's machine or the volume of the traffic is very low (only the automated behavior of the machine, e.g. periodic updates). However, during the working hours its network activity increases rapidly and it fluctuates during the working hours. For example during the lunch break there is a drop in the volume of the network traffic followed by a spike when users return to work. Note that the profile of the user's behavior changes through the day as user uses different services at different time of day. The model described in previous section does not reflect such changes and thus the quality of the generated data is lower.

Third problem that we have to consider is the *sequential character of the user's behavior*. The typical example is the e-mail usage. At first, user's e-mail client has to resolve the domain name of the IMAP server which will be seen in NetFlows as communication with the DNS server on port 53 over UDP protocol. Next, it synchronizes the e-mails in user's folders—it is represented as opening connection to the IMAP server on port 143 over TCP protocol. In the received e-mails there can be an interesting link that the user clicks to visit. This will appear as request to the DNS server that resolves the domain name from the link followed by a number of different connections to port 80 over TCP protocol to the HTTP server. From this example we can see that users exhibit sequential behavior and thus probability of two consequential services is not independent.

4.1 Model structure

In order to address all the problems discussed above, we propose a model (see Figure 2) composed from three components, all parametrized by day time.

The first component—*user model*—describes different aspects of the user's behavior, such as the timings between requests (*thinking time*), which *service* will he use and which *server/target* will he contact, its identity, etc. Note that the user does not necessary have to be human but some automated agent in the operating system as well (e.g. automatic updates).

The second component—the *system model*—models the automatic behavior of the user's operating system. This component models the process of assigning client ports.

The last component—*service model*—models the behavior of the remote service contacted by the user. This includes the amount of data transferred between client and server (bytes and packets as well), the duration of the response, the delay of the server and TCP flags (if the connection uses TCP protocol).

In order to simplify interactions between components and its internal models we adopt following assumptions:

a) The thinking time (T) depends only on the daytime and not on other aspects of user's behavior,

b) the client port ($cPort$) depends only on the service and day time—source ports for the outgoing connections are assigned by the operating system without any user' interaction. However, most operating systems simply increment the last used port until the range is depleted and thus for different daytime different range of ephemeral ports is used. The dependency on service is caused mainly by long persistent connections that are split by the NetFlow probe into several NetFlow records. All these flows have the same client port and in the statistics it appears as the service prefers single source port.

The negative impact of the assumption is that proposed model is not able to correctly capture the periodical behavior of particular service, however, it is outweighed by the benefit of the simplification of relations and internal models of individual components. The relaxation of adopted assumptions is left to future work.

4.2 Individual model components

The simplified interactions between internal models are following:

- *Thinking time*: Thinking time does not have any interaction or dependency on other components (see Assumption a)).
- *Service and target*: Target contacted by the user depends on the service requested by the user. This corresponds to the fact that there are specialized servers that serves only some (or even a single) services (HTTP server, database server, etc).
- *Client port*: Client port depends only on the service requested by the user (see Assumption b)).
- *Remote service model*: Model of remote depends on the service and target contacted by the user. For example different services on different servers has different profile (volume of the network traffic, server delay, content type, etc.)

In the following we will discuss internal models of each component separately.

Table 3: Set of features that describe the behavior of a service. Note that TCP flags for request and response are empty for all non-TCP connections.

Feature name	Description
Request bytes	Number of bytes in request
Request packets	Number of packets in request
Request protocol	TCP, UDP, ICMP, etc.
Request flags	TCP flags in request
Request length	Duration of request
Server thinking time	Time difference between request and response
Response bytes	Number of bytes in response
Response packets	Number of packets in response
Response length	Duration of response
Response flags	TCP flags in response
Has response	Is there corresponding response to the request?

4.2.1 *Thinking time*

The approach with marginal model we have proposed, models the starting time of the request as a sum of the starting time of the previous request and the thinking time of the client. However, such approach cannot be directly adopted in this model due to the fact that if we parametrize the thinking time with the daytime, it does not capture the time intervals when the activity of the user is very low. This causes artifacts that results in unrealistic data. For example, when first request occurs at 09:35 and next occurs at 11:20 we cannot compute the thinking time of flows between 10:00 and 11:00.

To avoid this issue, we do not estimate the thinking time directly. Instead we model the number of request n generated by user in given time interval[3]. The thinking time is then computed as follows

$$T = \frac{L}{n} \tag{3}$$

where L is the duration of usual time interval in seconds.

To denoise the input data—number of requests $[n_1, \dots, n_i, \dots]$ in five minute time window—we smooth the data with sliding window as follows

$$n_t = \frac{1}{l} \sum_{i=t-\frac{l}{2}}^{t+\frac{l}{2}} n_i, \tag{4}$$

where l is the width of the sliding window. It controls the smoothness of the estimate—if the window is too long, the value does not follow the trends in the data, and if it is too short the estimated value is too noisy.

Next, we divide the list of the number of requests $N = [n_1, \dots, n_i, \dots]$ into one-day long sets forming matrix N' defined in Equation 5. For every time interval $t \in [1, \dots, 288]$, that is represented by a row in matrix N', we have k samples where k is the length of the training data in days.

$$N' = \underbrace{\begin{pmatrix} n_1 & n_{289} & \cdots \\ n_2 & n_{290} & \cdots \\ \vdots & \vdots & \\ n_{288} & n_{576} & \cdots \end{pmatrix}}_{k} \tag{5}$$

We estimate the distribution of number of requests for interval t with histogram $\mathcal{H}_t$ with non-linearly distributed bins $[1, 11), [11, 21), [21, 41), [41, 81), [81, 201), [201, \infty)$. Furthermore, for every bin of the histogram $\mathcal{H}_t$ we define distribution of values. If the number of samples that fit into this bin is 1, we assume uniform distribution of values in this particular bin. If there is more samples, we assume normal distribution with parameters estimated from the samples that fit into the bin. This setup helps us to overcome the lack of data as we have only k samples.

The sampling procedure of the thinking time for given time interval is listed in Algorithm 1.

Algorithm 1 Sampling of thinking time

```
 1: procedure THINKINGTIME($t$) ▷ Sampling thinking time
 2:                                for time interval $t$
 3:     $b \sim \mathcal{H}_t$  ▷ Select bin with respect to distribution $\mathcal{H}_t$
 4:     if $|b| = 1$ then ▷ If there is only one training sample
 5:                        that fits in the bin $b$
 6:         $n_t \sim \mathcal{U}(b_a, b_b)$  ▷ Sample from uniform dist.
 7:                        defined by boundaries
 8:                        $b_a$ and $b_b$ of the bin $b$
 9:     else
10:         $n_t \sim \mathcal{N}(\mu, \sigma)$ ▷ Sample from normal distribution
11:                        with parameters estimated from
12:                        training samples
13:     end if
14:     $T = L/n_t$ ▷ Compute thinking time, $L$ is the length
15:                 of the time interval (in our case $L = 300$)
16:     return $T$
17: end procedure
```

4.2.2 *Service and target*

One of the problems of the models described in previous sections was the inability to precisely model the sequential character of the user's behavior. To address this issue we separate the probability of the service defined by following equation

$$p(s|t) \tag{6}$$

where $s \in S$ represents the service (the server port and protocol tuple) requested by the user and h is the day time. The sequential character is naturally modeled by *Markov chain* [19].

Next, we have to discuss model that describes which target will be contacted by the user. In Section 4 we have defined that the target contacted by the user depends on requested service and the day time. This is summarized into following probability

$$p(\mathrm{dIP}|s, t) \tag{7}$$

where dIP is the destination IP of contacted server (target), s is the requested service and t is the day time.

[3]The length of the time interval is 5 minutes—the usual length of the batch in anomaly detection [17, 18].

4.2.3 Client port

As we have discussed above, in order to generate NetFlow data we can simplify the operating system model to model only the client ports. In assumption b) we stated that client port depends on the service and the daytime. Using this assumption we can model the assigning of the ephemeral ports by the probability defined as follows

$$p(\text{cPort}|s, t) \tag{8}$$

where cPort represents the client port assigned by the operating system and service s. This model captures the long term connections to a service that appear in the data as different NetFlows with the same client port (connection to e-mail server, long term SSH connection, etc.) as well as different strategies used by operating system to assign the ephemeral source port to outgoing connections.

4.2.4 Remote service model

Internal models that we have discussed in previous paragraphs addressed only single feature of the component. However in order to capture the relations between individual NetFlow features we model the whole component together by single joint probability defined as follows

$$p(x_s|\text{dIP}, s, t) \tag{9}$$

where $x_s \in X_s$ represents the space defined by features of the service listed in Table 3, dIP is the contacted server, s is the requested service and t is the day time. Using this model we can precisely capture the behavior of the service. However, such model captures precisely the behavior that appeared in the data and it is not able to generate completely new values. To overcome this issue we can add the gaussian noise to the Equation 9 and thus generate new previously unseen data. The strength of the noise λ is the parameter of the model and allows us to control how "realistic" the generated data should be.

Before we generate new flows we train the model as described in Sections 4.2.1, 4.2.2 and 4.2.4. Next, we simulate the data as described in Algorithm 2.

Algorithm 2 Sampling of the NetFlow data

```
1: procedure SAMPLEFLOW(length)                ▷ Sampling single flow
2:     $\mathcal{F} \leftarrow \emptyset$
3:     $t \leftarrow 0$                               ▷ Set current time to 0
4:     repeat
5:         $T \leftarrow thinkingTime(t)$             ▷ Sample thinking time
6:         $s \leftarrow p(s|t)$                      ▷ Sample service
7:         $\text{dIP} \leftarrow p(\text{dIP}|s,t)$   ▷ Sample target
8:         $cPort \leftarrow p(\text{cPort}|s,t)$     ▷ Sample client port
9:         $x_s \leftarrow p(x_s|\text{dIP},s,t)$     ▷ Sample remaining features
10:        $\text{flow} \leftarrow t, s, \text{dIP}, \text{cPort}, x_s$   ▷ Build flow
11:        $\mathcal{F} \leftarrow \mathcal{F} \cup \{\text{flow}\}$
12:        $t \leftarrow t + T$                       ▷ Increment current time
13:    until $t \leq$ length
14:    return $\mathcal{F}$
15: end procedure
```

4.3 Possible extensions

In previous paragraphs we have discussed complex solution for generating NetFlow data. However, the general

Table 4: Capabilities of presented models. The table summarizes capabilities of individual models. It shows wheter given property of the traffic can be (✓), can not be (✗) or can be with specific settings ($\frac{1}{2}$) captured by given model. Note that Time variant joint probability model is able to capture the sequential character of the user's behavior when the service model is replaced by Markov chain.

	Random sampling (Section 3.1)	Sampling with marginal model (Section 3.2)	Time variant joint model (Section 4)
Properties of fields of NetFlows (e.g. distribution of bytes, source ports, etc.)	✗	✓	✓
Intraflow relations (e.g. Packet/Bytes ration, etc.)	✗	✗	✓
Interflow relations (e.g. request/response ration)	✗	✓	✓
Changes in the user's behavior	✗	✗	✓
Sequential character of the user's behavior	✗	✗	$\frac{1}{2}$

schema of the proposed model can be extended to capture not only NetFlow data but different types of communication as well. Note also that the the *system model* can be extended to model installed applications. It can affect the number of connection to the server, duration of the request and the thinking time of the server (different versions of an internet browser uses different number of concurrent connections). Another component that can be extended is the *service model*. As it controls the behavior of the remote service it is natural to extend it with application specific features such as URL or content type. However, these extensions, though beneficial, are out of the scope of this paper and will be considered in future work.

5. EVALUATION

In this section we evaluate the quality of the data generated by individual sampling approaches defined in Section 3. We have implemented a set of state-of-the-art detection algorithms to evaluate the simulated data and compared this data with real-world traffic. Using the *Jensen-Shannon divergence* (JSD) [20] we then measure the distance between distribution of anomaly values of real-world and artificially generated data.

5.1 Selected anomaly detection algorithms

The goal of presented model is to generate data for evaluation of an anomaly detection algorithm. Therefore, we have implemented various types of algorithms based on different detection paradigms. This allows us to measure the quality of the generated data under different conditions.

Algorithms proposed by Pevný et al. [21] and Lakhina et al. [22, 23] use the principal component analysis to detect anomalies in the traffic. However, there are several key differences between these methods. First difference is in the features that are used for the definition of the model of the individual detectors. Second difference is the measure used for assigning the anomaly value (Lakhina proposes to use *reconstruction error* and Pevný uses *mahalanobis distance in sub-spaces*). Note that we have implemented four different versions of algorithm proposed by Pevný denoted in the results as *Pevný-f-dIP*, *Pevný-f-sIP*, *Pevný-$f^{\perp}$-dIP* and *Pevný-$f^{\perp}$-sIP* (all described in [21]), and two versions of Lakhina's algorithm where version listed in the results as *Lak. Vol.-sIP* models the traffic with respect to a source IP and version denoted as *Lak. Vol.-dIP* models the traffic with respect to a destination IP.

Second type of algorithm that we have implemented is a modified version of *Minnesota Intrusion Detection System–MINDS* [24]. It uses an internal model of the network traffic but unlike the algorithms proposed by Pevný and Lakhina it does not uses the PCA but measures the difference between last and current time window. In order to overcome the performance issues we have modified the algorithm from the originally proposed version. The modifications are described in [25].

The last group of algorithms does not use any internal model of the network traffic. Method originally published by Kuai Xu *et. al.* [26] uses basic assumption that all network traffic could be classified into several categories using set of static thresholds. In addition to the original algorithm (denoted as *Xu-sIP* in out evaluation) we have implemented modified version (denoted as *Xu-dIP*) that uses complementary features relating to the destination IP.

5.2 Training and evaluation data

To evaluate the quality of the simulated traffic we have used the data recorded on university campus during the one week in April 2013 (further denoted as $\mathcal{D}_{\text{orig}}$). From the recorded data we have selected a set of full-time employees with various user profiles (developers, scientists, managers and administrative staff). We have separated their traffic based on the IP address of selected users and use it as training data for two models defined in Sections 3.2 and 4. The remaining traffic formed the reduced dataset $\mathcal{D}_{\text{red}}$ and was used as background that was mixed with the simulated traffic.

The evaluation of quality of the generated data was separated into two stages. During the first stage we processed the original data $\mathcal{D}_{\text{orig}}$ separately by all anomaly detection methods and for every detection method we estimated the distribution of anomaly values of selected users. These distributions then served in the comparison as a baseline.

In the second stage we simulated the user's behavior using four approaches proposed in this paper: (1) the random sampling (referred as *Random*, see Section 3.1), (2) sampling with marginal model (*Marginal*, see Section 3.2), (3) sampling with time variant joint probability model (*Model*, see Section 4) and (4) sampling with time variant joint probability model with additional noise that affected the sampled data (*ModelN*, strength of the noise $\lambda = 10^{-5}$). The generated data were mixed with the reduced dataset $\mathcal{D}_{\text{red}}$ and separately processed by all anomaly detection algorithms. Next, for every detection algorithm we have again estimated the distribution of the anomaly values of the simulated traffic and measure the value of JSD between the distribution of the simulated traffic and real traffic. This process was repeated 20 times. Next we have used Kruskal-Wallis statistical test on significance level $\alpha = 0.05$ to detect whether the results for different simulation approaches are significantly different or not. The test proved that the results for different simulation approaches are different enough to compare only the mean values.

5.3 Quality of the generated data

The results are summarized in Table 5. It shows that the random sampling generates the least realistic data. The value of JSD is by order of magnitude larger compared to the two remaining models. This confirms the expectations that the random sampling can be used only for syntax testing as we have discussed in Section 3.1. The second approach, the sampling with marginal model, provides significantly better results compared to the random sampling. The results show that correct estimation of the marginal distribution of individual features improves the results by order of magnitude. However, assumptions that (1) all inter-flow features are independent and (2) user's behavior does not depend on the day time clearly do not hold. Therefore, the most advanced approach, the sampling from the time variant joint probability model, provides the results on average 2.3× better than sampling with marginal model. The last approach shows that by adding the low volume of Gaussian noise into the model we can generate data that are completely new but still follow the original user profiles. Such approach is important for testing the detection boundaries of the detection algorithms.

The last results shown in Figures 3 and 4 visualize the distributions of anomaly value of real data and data sim-

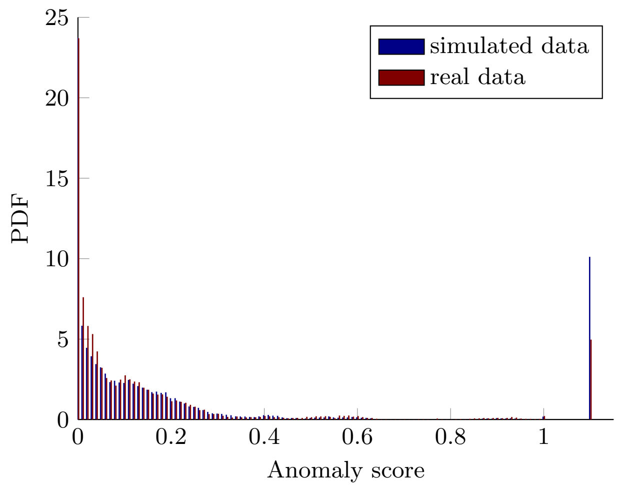


Figure 3: Distribution of anomaly values of Pevný-f-dIP method for the real traffic and traffic simulated by the time variant joint probability model.

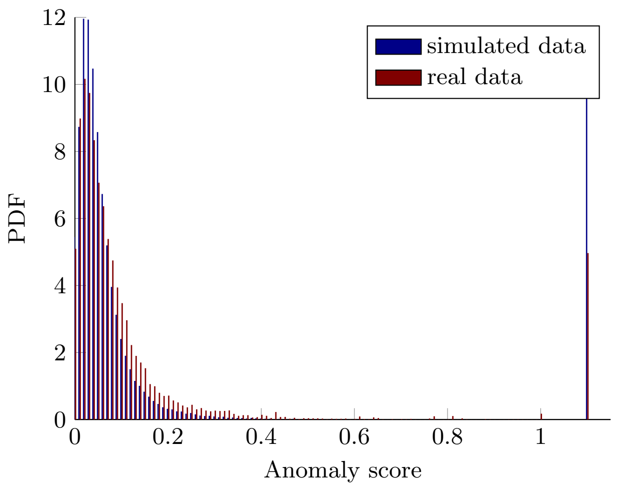


Figure 4: Distribution of anomaly values of Pevný-$f^{\perp}$-dIP method for the real traffic and traffic simulated by the time variant joint probability model.

Table 5: Jensen-Shannon divergence between distribution of anomaly values of real and simulated traffic.

Detection alg.	Model	Marginal	Random	ModelN
Pevný-f-dIP [21]	0.0133	0.0324	0.4596	0.0128
Pevný-f-sIP [21]	0.0146	0.0312	0.4780	0.0122
Pevný-$f^{\perp}$-dIP [21]	0.0167	0.0322	0.4675	0.0138
Pevný-$f^{\perp}$-sIP [21]	0.0175	0.0320	0.4519	0.0130
Lak.Ent. [23]	0.0413	0.0906	0.1198	0.0414
Lak.Vol.-sIP [22]	0.0199	0.0756	0.1076	0.0211
Lak.Vol.-dIP [22]	0.0241	0.0670	0.0938	0.0250
MINDS [24]	0.0184	0.0557	0.1703	0.0170
Xu-sIP [26]	0.0172	0.0188	0.0908	0.0172
Xu-dIP [26]	0.0193	0.0405	0.2507	0.0183
Average	0.0202	0.0476	0.2690	0.0192

ulated by time variant joint probability model. Note that anomaly score 1.1 represents the flows where the particular detector provided no results (due to its limitations). These figures show that our model generates traffic that triggers response of anomaly detection algorithms practically indistinguishable from the response to real traffic.

6. CONCLUSION

We proposed a solution for generating realistic NetFlow data that can be used for evaluation and configuration of anomaly detectors. We introduced the *time variant joint probability model* that is able to capture inter- and intra-flow relations as well as sequential character of user's behavior. We compared the proposed solution with two other simpler models (*random sampling* and *sampling with marginal model*) and have shown that our solution provides more than 2.3× better.

In future work we will focus on relaxing the assumptions adopted in this paper that limits the quality of the generated data. We will extend the general schema to capture the application specific aspects of the network traffic. The second issue that we will address is the modeling of the whole actions and not individual flows that will enable us to correctly model for example the full load of the HTML page.

7. ACKNOWLEDGMENT

The work presented in this paper was supported by Ministry of the Interior project VG20122014079. The work of T. Pevný was also supported by the Grant Agency of Czech Republic under the project P103/12/P514.

REFERENCES

[1] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, Jul. 2009. [Online]. Available: http://dl.acm.org/citation.cfm?id=1541880.1541882 http://portal.acm.org/citation.cfm?doid=1541880.1541882

[2] A. Patcha and J.-M. Park, "An overview of anomaly detection techniques: Existing solutions and latest technological trends," *Computer Networks*, vol. 51, no. 12, pp. 3448–3470, Aug. 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S13891286070

[3] P. Barford and D. Plonka, "Characteristics of network traffic flow anomalies," *Proceedings of the 1st ACM SIGCOMM Workshop ...*, 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=505211

[4] "The network simulatorâĂŞNS-2," http://nsnam.isi.edu/nsnam, [Online; accessed June 20th, 2012].

[5] T. Henderson and M. Lacage, "Network simulations with the ns-3 simulator," *SIGCOMM*, p. 2006, 2008. [Online]. Available: http://conferences.sigcomm.org/sigcomm/2008/papers/p527-hendersonA.pdf

[6] A. Varga and R. Hornig, "AN OVERVIEW OF THE OMNeT++ SIMULATION ENVIRONMENT," *Proceedings of the First International ICST Conference on Simulation Tools and Techniques for Communications Networks and Systems*, 2008.

[Online]. Available: http://eudl.eu/doi/10.4108/ICST.SIMUTOOLS2008.3027

[7] R. Bye, S. Schmidt, K. Luther, and S. Albayrak, "Application-level simulation for network security," *Proceedings of the First International ICST Conference on Simulation Tools and Techniques for Communications, Networks and Systems*, 2008. [Online]. Available: http://eudl.eu/?id=2961

[8] D. Grunewald, R. Bye, K. Bsufka, and S. Albayrak, "Agent-based Network Security Simulation (Demonstration)," *AAMAS*, pp. 1325–1326, 2011.

[9] L. M. Rossey, J. C. Rabek, R. K. Cunningham, D. J. Fried, R. P. Lippmann, and M. A. Zissman, "LARIAT : Lincoln Adaptable Real-time Information Assurance Testbed," pp. 1–27, 2001.

[10] A. Sperotto, R. Sadre, P.-t. D. Boer, and A. Pras, "Hidden Markov Model modeling of SSH brute-force attacks," *9th IEEE International Workshop on IP Operations and Management (IPOM 09)*, p. 13, 2009.

[11] D. Brauckhoff and A. Wagner, "FLAME: a flow-level anomaly modeling engine," in *The conference on Cyber security*, 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1496663

[12] P. Siska, M. P. Stoecklin, A. Kind, and T. Braun, "A flow trace generator using graph-based traffic classification techniques," *Proceedings of the 6th International Wireless Communications and Mobile Computing Conference on ZZZ - IWCMC '10*, p. 457, 2010. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1815396.1815503

[13] J. Sonchack and A. Aviv, "LESS Is More: Host-Agent Based Simulator for Large-Scale Evaluation of Security Systems," *Computer Security-ESORICS 2014*, pp. 365–382, 2014. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-11212-1_21

[14] S. Floyd and V. Paxson, "Difficulties in simulating the Internet," *IEEE/ACM Transactions on Networking (TON)*, vol. 9, no. 4, pp. 392–403, 2001. [Online]. Available: http://dl.acm.org/citation.cfm?id=504642

[15] H. Ringberg, M. Roughan, and J. Rexford, "The need for simulation in evaluating anomaly detectors," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 1, p. 55, Jan. 2008. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1341431.1341443

[16] J. Sommers, H. Kim, and P. Barford, "Harpoon: a flow-level traffic generator for router and network tests," *ACM SIGMETRICS Performance . . .* , pp. 392–393, 2004. [Online]. Available: http://dl.acm.org/citation.cfm?id=1005733

[17] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan, "Network anomography," p. 30, Oct. 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251086.1251116

[18] F. Silveira, C. Diot, N. Taft, and R. Govindan, "ASTUTE: Detecting a different class of traffic anomalies," *ACM SIGCOMM Computer . . .* , 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1851215

[19] K. Murphy, *Machine Learning: a Probabilistic Perspective*, 2012.

[20] D. Endres and J. Schindelin, "A new metric for probability distributions," *IEEE Transactions on Information Theory*, vol. 49, no. 7, pp. 1858–1860, Jul. 2003. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber

[21] T. Pevny, M. Rehak, and M. Grill, "Detecting anomalous network hosts by means of PCA," in *2012 IEEE International Workshop on Information Forensics and Security (WIFS)*. IEEE, Dec. 2012, pp. 103–108. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber

[22] A. Lakhina, M. Crovella, and C. Diot, "Diagnosing network-wide traffic anomalies," *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications - SIGCOMM '04*, p. 219, 2004. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1015467.1015492

[23] ——, "Mining anomalies using traffic feature distributions," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 4, p. 217, Oct. 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1090191.1080118 http://dl.acm.org/citation.cfm?id=1080118

[24] L. Ertoz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, and P. Dokas, "Minds-minnesota intrusion detection system," *Next Generation Data Mining*, 2004. [Online]. Available: http://www.it.iitb.ac.in/ deepak/deepak/courses/mtp/papers/m minnesota intrusion detection system.pdf

[25] M. Rehak, M. Pechoucek, K. Bartos, M. Grill, and P. Celeda, "Network Intrusion Detection by Means of Community of Trusting Agents," *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT'07)*, pp. 498–504, Nov. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber

[26] K. Xu, Z. Zhang, and S. Bhattacharyya, "Reducing unwanted traffic in a backbone network," *Usenix Workshop on Steps to Reduce Un- wanted Traffic in the Internet (SRUTI 05)*, 2005. [Online]. Available: https://www.usenix.org/event/sruti05/tech/talks/xu.pdf